A Multiscale Approximation Algorithm for the Cardinality Constrained Knapsack Problem

by

Bharath Kumar Krishnan

B.Tech. Civil Engineering
Indian Institute of Technology Madras, 1999

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author .....
Department of Civil and Environmental Engineering
February 13, 2006

Certified by .....
George A. Kocur
Senior Lecturer, Department of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .....
Andrew J. Whittle
Chairman, Department Committee for Graduate Students

A Multiscale Approximation Algorithm for the Cardinality Constrained Knapsack Problem

by

Bharath Kumar Krishnan

Submitted to the Department of Civil and Environmental Engineering
on February 13, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computational Engineering

Abstract

I develop a multiscale approximation algorithm for the cardinality constrained knapsack problem. The algorithm consists of three steps: a rounding and reduction step where a hierarchical representation of the problem data ranging from coarse to fine is generated, a solution step where a coarse solution is computed, and a refinement step where the accuracy of the solution is improved by refining the problem representation.

I demonstrate that the algorithm is fully polynomial with a runtime complexity that improves upon the previous best known fully polynomial approximation scheme. Through an extensive computational study, I show that the running times of the algorithm is less than or equal to that of a commercial integer programming package with little loss in solution accuracy.

Thesis Supervisor: George A. Kocur
Title: Senior Lecturer, Department of Civil and Environmental Engineering

Acknowledgments

I would like to thank my advisor, Prof. Kevin Amaratunga, for his guidance, support and encouragement throughout my stay at MIT. Kevin's patience, flexibility and pragmatism allowed me to pursue areas of research that interested me while keeping in sight my ultimate goals.

I would like to thank Prof. George Kocur for agreeing to be my thesis committee chairman. Without his incredible support and guidance, this thesis would not have been possible. His constant help and encouragement helped me crystallize the raw ideas and results I had into the finished product.

I would like to thank Prof. Steven Lerman for his attention to detail while reading my dissertation. His questions, suggestions and practical advice were of invaluable help.

I would also like to thank Prof. Rory O'Connor, Prof. Patrick Jaillet and Prof. Nitin Patel for their help and advice in my search for a research problem.

I would like to thank Prof. Lerman, Dr. Jud Harward, Prof. Kocur and the 1.00 TAs for providing me a wonderful environment to work in as a teaching assistant. Course 1.00 facilitated my stay at MIT and gave me the independence to pursue areas of study that satisfied my intellectual curiosity.

Sudarshan, Sugata, Stefan, Scott, George, Petros, Ying-Jui, Julio and Joan McCusker made IESL a wonderful environment to work in. My friends at MIT made it a wonderful place to live in.

I would like to thank Raji Athai and Anantha Athimber, for their help, support and the confidence they showed in me throughout my undergraduate and graduate studies. I would like to thank Amma, Appa and my sister Priya for their love, support and encouragement.

And finally, I thank my wife and best friend, Asti, without whose love, sensibility and support this thesis would not have been possible.

Contents

List of Figures

List of Tables

1

Introduction

This thesis develops an approximation algorithm for the cardinality constrained binary knapsack problem. A cardinality constraint is a restriction on the number of items included in the optimal solution to a knapsack problem. This restriction naturally arises in cases where external costs in proportion to the number of items selected have to be considered as part of the solution. In this chapter, I give an overview of the problem and a short description of existing solution methodologies. I discuss the limitations of existing approximation algorithms and motivate the multiscale approximation approach. I close with this thesis's contributions and the structure of the dissertation.

1.1 The Binary Knapsack Problem

The Binary Knapsack Problem (BKP) is the simplest non-trivial integer programming problem. BKP and its variations have been widely studied in the past, see for example, Martello and Toth [17], Kellerer et al [12].

The problem is stated as follows:

Given N items, each with a profit p_i and weight w_i, and a knapsack with capacity c, our objective is to select as many items as possible such that the items fit inside the knapsack while maximizing our profit. We can formulate BKP as an integer programming problem as follows.

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} p_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i x_i \leq c \\ & x_i \in \{0,1\} \end{aligned} \tag{1.1}$$

Often, a natural restriction that arises is a constraint on the number of items that can be part of the solution. If k is an upper bound on the number of items that can be selected, we can write the resulting cardinality constrained knapsack problem (kKP) as follows.

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} p_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i x_i \leq c \\ & \sum_{i=1}^{N} x_i \leq k \\ & x_i \in \{0,1\} \end{aligned} \tag{1.2}$$

where p_i, w_i, $c \in \Re^+$ and $N \gg k \geq 1$.

The kKP is a special case of a two constraint knapsack problem that generalizes the BKP. Both the BKP and kKP belong to the family of $\mathcal{NP}$-hard problems

(for example, see Gary and Johnson [6], Papadimitriou and Steiglitz [21]). There is currently no known polynomial time algorithm for the kKP. A variation of kKP is the Exact-kKP where exactly k items have to be selected in the solution. Caprara et al [3] show that this version of the problem easily reduces to the kKP.

1.2 Approximation Algorithms

BKP and kKP are combinatorial in nature as the feasible solutions are subsets of a given set. We can solve these problems exactly by enumeration. However, this is not always practical as the number of feasible solutions is typically exponential in the size of the problem. The BKP allows a pseudopolynomial approximation to any required degree (for example, see Vazirani [28]). In 2000, Caprara et al [3] developed the first pseudopolynomial approximation to the kKP.

1.2.1 Definitions

Let Π be an instance of a maximization problem and let $\mathcal{A}$ be an approximation algorithm. Let $OPT(\Pi)$ be the optimal value of the objective function for the problem and $\mathcal{A}(\Pi)$ be the objective function value achieved by algorithm $\mathcal{A}$. For a given error value ϵ, we say that $\mathcal{A}$ is an $(1-\epsilon)$-approximate scheme if

$$\mathcal{A}(\Pi) \geq (1-\epsilon)OPT(\Pi) \tag{1.3}$$

$\mathcal{A}$ is also said to have a performance ratio of ϵ. Furthermore, if $\mathcal{A}$ is polynomial in the size of Π, we call $\mathcal{A}$ a *Polynomial Time Approximation Scheme*($PTAS$). The complexity of a $PTAS$ is often exponential in $\frac{1}{\epsilon}$. If $\mathcal{A}$ is polynomial in the size of the problem and $\frac{1}{\epsilon}$, we say that $\mathcal{A}$ is a *Fully Polynomial Time Approximation Scheme* ($FPTAS$). An $FPTAS$ is the strongest possible algorithm with respect to worst case complexity (see Schuurman and Woeginger [26]) that we can devise for an $\mathcal{NP}$-hard problem.

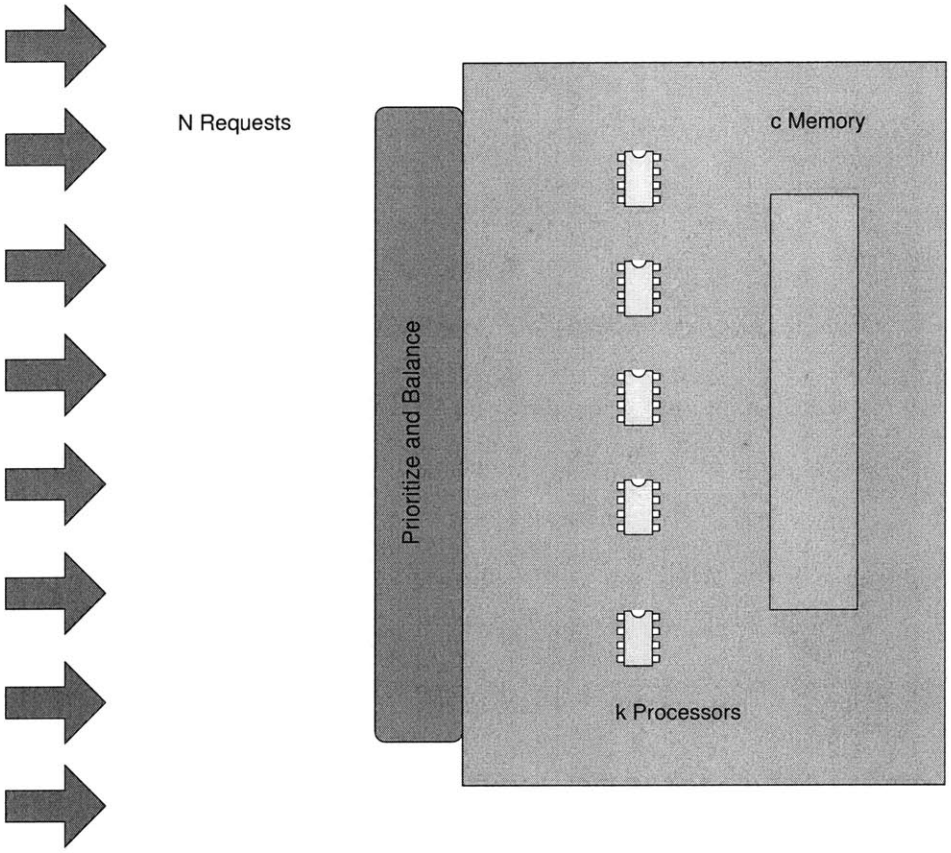


Figure 1-1: Scheduling tasks on a multi-processor shared memory computer

1.3 Literature Review

1.3.1 Examples of the kKP

Consider the following resource allocation problem:

An application server has resource constraints in the form of the number of processors and the amount of core memory (Figure 1-1). At a given time, a large number of processes are competing for these resources. Without loss of generality, assume that each process requires one dedicated processor and some amount of memory. Every process also has a priority or profit value attached to it. The problem here is to optimize the selection of processes to run at a given time so as to maximize the utility (or profit) and make full use of the processor and memory resources.

The kKP appears as a subproblem (see for example. Caprara et al [3]) in a

wide variety of instances. Posner and Guignard [24] present the Collapsing Knapsack Problem that can be solved as multiple kKP instances with the same set of items and different capacities. This problem has applications in satellite communications, where each user is allocated a portion of the transmission band and gaps have to be maintained between each portion.

The kKP appears as a subproblem when Cutting Stock problems are solved using column generation under the condition that the number of pieces that can be cut from each stock is constrained due to the limited number of cutting knives available(see for example, Kolen and Spieksma [13], Vanderbeck [27]).

Farias and Nemhauser [4] study a cardinality constrained knapsack problem with continuous variables, where no more than a specific number of variables are allowed to be positive. In this variation, fractional items can be selected. They cite applications of this variation in areas such as finance, location and scheduling.

Cardinality constraints occur naturally in a variety of situations. For example, in portfolio optimization, a practical consideration is the number of stocks that are held as part of the portfolio. In assortment space optimization where retail shelf space is the primary constraint, secondary constraints in terms of the number of brands carried per shelf appear. In some professional sports, the number of players a team can carry and the total salary that they can be paid per year are both limited.

1.3.2 Approximation Algorithms for the kKP

Dynamic Programming is a technique to efficiently enumerate the search space for a combinatorial problem. It is guaranteed (for example, see Woeginger [29]) to find the exact optimal solution, but not necessarily in polynomial time. A traditional way to derive an $FPTAS$ from a Dynamic Programming solution to an $\mathcal{NP}$-Hard problem is to round the input data [25].

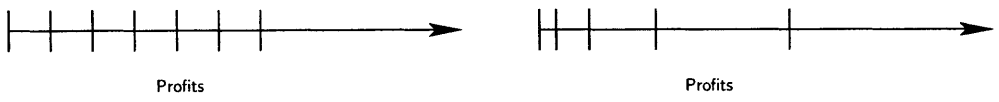
Korte and Schrader [14] proved that a knapsack problem with two or more constraints allows a $FPTAS$ only if $\mathcal{P} = \mathcal{NP}$. The kKP is of theoretical interest as it is a special case of a two dimensional knapsack problem.

Caprara et al [3] proved the existence of a $FPTAS$ for the kKP. In this algorithm,

the item profit values are scaled uniformly. Using a dynamic programming scheme, a heuristic solution that is optimal for the scaled profits is obtained. This solution is then showed to be within $(1 - \epsilon)$ of the optimal solution to the original kKP.

The complexity of this approach is $O(\frac{Nk^2}{\epsilon})$ where N is the number of items, k is the cardinality constraint and ϵ is the target approximation error.

The complexity of the dynamic programming solution depends upon the number of distinct profit values (represented by the number of bins) in the scaled problem. Uniformly scaled item profits form an arithmetic series of bins (Figure 1-2(a)). An alternate approach is to construct a series of bins on the profit axis that increase in size geometrically as seen in Figure 1-2(b).



(a) Arithmetic Bins (b) Geometric Bins

Figure 1-2: Arithmetic and Geometric Bins

Geometric binning reduces the number of distinct profit values; however this leads to a dynamic program with exponential complexity [18]. Hutter and Mastrolilli [18] propose a novel hybrid rounding scheme for item profits that combines both arithmetic and geometric rounding. In this algorithm, the items are divided into two sets based on their profits. A combination of arithmetic and geometric bins are then used to scale the item profit values. A dynamic program similar to that in [3] is used to compute combinations of large profit items with a given total profit and minimum weight. These combinations are then augmented using the small profit items. The Mastrolilli-Hutter algorithm has a complexity of $O(N + \frac{kz^2}{\epsilon^2})$ where $z = \min\{k, \frac{1}{\epsilon}\}$, with a performance ratio of ϵ.

In general, N is considered to be much larger in magnitude than $\frac{1}{\epsilon}$ [15, 12]. The Mastrolilli-Hutter algorithm has an advantage as it is explicitly linear in N when compared to the earlier algorithm of Caprara et al.

1.3.3 Approximation Algorithms for the BKP

The Binary Knapsack Problem (BKP) is a special case of the kKP with no constraint on the number of items that can be selected, i.e., $k = N$. Scaling the item profits followed by a dynamic programming solution is a common technique in devising fully polynomial approximation schemes for the BKP.

The first fully polynomial approximation scheme for the BKP was discovered independently by Ibarra and Kim [8] and Babat [1]. In this algorithm, items are divided into two sets according to their profits. The item profits are scaled and the knapsack is solved for the large profit items using dynamic programming. Any residual capacity in the list of dynamic program solutions is filled by the small profit items. This algorithm has a time complexity of $O(N \ln N + \frac{1}{\epsilon^2} \min\{N, \frac{1}{\epsilon^2} \ln(\frac{1}{\epsilon})\})$ with a space complexity of $O(N + \frac{1}{\epsilon^3})$.

Using a different scaling technique, Lawler [15] improved the time complexity of the Ibarra-Kim algorithm to $O(N \ln \frac{1}{\epsilon} + \frac{1}{\epsilon^4})$. Magazine and Oguz [16] modified the Ibarra-Kim algorithm to provide another $FPTAS$ for the BKP that has weaker time complexity ($O(\frac{N^2 \ln N}{\epsilon})$) but a stronger space complexity ($O(\frac{N}{\epsilon})$) when ϵ is appropriately chosen.

Kellerer and Pferschy [10] describe a $FPTAS$ for the BKP with a time complexity of $O(N \min\{\ln N, \ln \frac{1}{\epsilon}\} + \frac{1}{\epsilon^2} \ln \frac{1}{\epsilon} \min\{N, \frac{1}{\epsilon} \ln \frac{1}{\epsilon}\})$ and a space requirement of $O(N + \frac{1}{\epsilon^2})$. In the first step in this algorithm, items are separated into two sets depending on their profit values. The range of the large profit items is then subdivided uniformly into bins. Each bin is further subdivided into smaller intervals where the interval size depends on the bin profit value. Bins with larger profit values get a smaller number of subintervals. Within each bin, items are ordered according to weight and their profits are approximated by the lower subinterval bound. As before, a dynamic programming scheme is applied to the scaled profit values and solutions are computed. Small profit items are used in a greedy heuristic to fill in any excess capacity. This algorithm has improved space utilization when compared to the Magazine-Oguz [16] algorithm when $N \geq \frac{1}{\epsilon}$ and improved time complexity when $N \ln N \geq \frac{1}{\epsilon}$.

1.4 Motivation and Contributions

The Mastrolilli-Hutter algorithm is linear in the number of items N considered in the problem. This is significant as N is assumed to be the dominating factor for a typical kKP. However computational experiments (detailed in Chapter 4) show that this algorithm performs poorly when compared to a state of the art mixed integer program (MIP) solver like CPLEX 9.0 [9]. An algorithm with guaranteed error bounds and practical utility is thus desirable.

Current rounding approaches discretize the range of profit values that an item can take. The sizes of each of these discrete bins is independent of the distribution of items within the range of profits. This motivates the idea of adaptive rounding which adapts bin sizes to the item profit distribution. If this leads to a further reduction in the input data size, the complexity of an algorithm utilizing this approach should be improved.

In this thesis I develop one such adaptive rounding technique (`Split/Merge`). This technique uses a suitable starting point to construct a hierarchy of bins on the profit values. The hierarchy of bins corresponds to a multiscale approximation of the item profits with the error varying by the scale. I show that this leads to a $(1-\epsilon)$-approximate algorithm ($MSKP$) for the kKP with complexity $O(N + \frac{kz}{\epsilon^2})$ where $z = \min\{k, \frac{1}{\epsilon}\}$. I also compare the computational performance of the algorithm against CPLEX and show that for a variety of randomly generated problem sets, the running time for $MSKP$ is competitive with or better than that achieved by CPLEX with little loss in solution quality.

1.5 Dissertation Structure

Chapter 2 provides an overview of arithmetic, geometric and hybrid binning procedures and the application of dynamic programming to derive an $FPTAS$ for the kKP. I describe `Split/Merge`, the adaptive rounding scheme that generates a multiscale representation of the item profits and show how the hybrid rounding of Mastrolilli and

Hutter can serve as its starting point. Chapter 2 also outlines the $MSKP$ algorithm that uses the multiscale representation to find the optimal solution to the kKP.

Chapter 3 gives the complexity analysis of $MSKP$ for continuous and discrete distributions of item profits. I show how $MSKP$ leads to an improvement in complexity of $O(z)$ where $z = \min\{k, \frac{1}{\epsilon}\}$.

Chapter 4 demonstrates the practical utility of $MSKP$ by comparing its running times and solution quality to that of CPLEX, a commercial integer programming solver. I show that for a variety of problem instances, randomly generated as detailed in Pisinger [22], the running times of $MSKP$ match or do better than that of CPLEX with little loss in solution quality. Chapter 5 summarizes the thesis and indicates avenues for further research.

2

A Multiscale Approximation Algorithm

This chapter describes the building blocks of a multi-scale approximation algorithm ($MSKP$) for the k-Knapsack Problem (kKP). The algorithm consists of a pre-processing step where a hierarchical representation of the problem data is generated, a solution step where a coarse solution is computed, and a refinement step where the accuracy of the solution is improved by refining the problem representation.

2.1 The k-Knapsack Problem (kKP)

kKP can be formulated as follows.

$$\begin{aligned} \max \quad & \sum_{i=1}^{N} p_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i x_i \leq c \\ & \sum_{i=1}^{N} x_i \leq k \\ & x_i \in \{0,1\} \end{aligned}$$

where p_i, w_i, $c \in \Re^+$ and $N \gg k \geq 1$.

A general approximation strategy for simplifying the problem structure involves the following steps.

1. Reduce the number of distinct profit values in the problem by rounding/binning the profit values.

2. Split the set of items into large profit and small profit items.

3. Solve the resultant simpler problem for the large profit items using a pseudo-polynomial dynamic program.

4. Use the small profit items to fill in any excess knapsack capacity.

2.2 A $\frac{1}{2}$-Approximate Algorithm

Relaxing the integrality of x_i leads to a $\frac{1}{2}$-approximation algorithm for the kKP as shown in Caprara et al. [3]. This LP approximation to the kKP can be computed in $O(N)$ time using the algorithm of Meggido and Tamir [20]. Algorithm 1 lists the $\frac{1}{2}$-Approximate Algorithm to the kKP.

The $\frac{1}{2}$-approximate algorithm helps us derive some useful bounds for the optimal value.

Algorithm 1 The $\frac{1}{2}$-Approximate Algorithm

1: $I \leftarrow$ `LPSolve`(kKP) {Index Set of Optimal Basic Solution}
2: $I_1 \in I$ {Indices with $x_i = 1$}
3: $I_f \in I$ {Indices with fractional x_i}
4: **if** $I_f = \emptyset$ **then**
5: `return` $Z^H = P(I_1)$
6: **else if** $I_f = \{i\}, i \in N$ **then**
7: `return` $Z^H = \max(P(I_1), p_i)$
8: **else if** $I_f = \{i, j\} \in N,\ w_j \leq w_i$ **then**
9: `return` $Z^H = \max(P(I_1) + p_j, p_i)$
10: **end if**

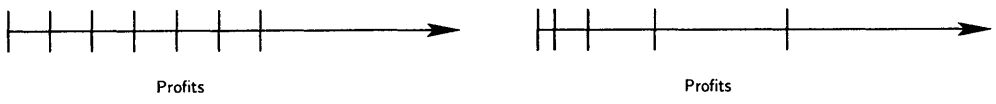
Let Z^H be the optimal solution obtained in this fashion. Let Z^* be the actual optimal profit sum. We have,

$$2Z^H \geq Z^H + p_{max} \geq Z^* \geq Z^H \tag{2.1}$$

where p_{max} is the maximum profit value of any item.

2.3 *FPTAS* for *kKP*

The first *FPTAS* for the *kKP* was shown in Caprara et al. [3] using uniformly sized bins (Fig. 2-1(a)), i.e., bin profits increasing in an arithmetic series, to round the profit values. This algorithm has a time complexity of $O(\frac{Nk^2}{\epsilon})$. The current best *FPTAS* is linear in N and is claimed by Mastrolilli and Hutter [19] using a hybrid rounding scheme. This scheme uses a combination of arithmetic and geometric progressions of bin sizes (Fig. 2-1(b)). Their algorithm has a running time of $O\left(N + \frac{kz^2}{\epsilon^2}\right)$ where $z = \min\{k, \frac{1}{\epsilon}\}$ with a performance ratio of ϵ.



(a) Arithmetic Bins (b) Geometric Bins

Figure 2-1: Arithmetic and Geometric Bins

2.4 A Multiscale Approximation

In a multiscale algorithm, the problem instance is represented across a range of scales. The algorithm solves the problem at a coarse scale and then refines the solution as needed to improve accuracy. To derive a multiscale algorithm for the kKP, we build a hierarchy of bins to approximate the item profits. Algorithm 2 shows the high level description of $MSKP$, the multiscale approximation scheme to the kKP.

Algorithm 2 The Multiscale Approximation Algorithm $MSKP$

1: $p' \leftarrow$ `SplitMerge`(p, τ, m)\{Adaptive Binning\}
2: $Z_0^* \leftarrow$ `Solve`(p', w, c)\{Dynamic Program\}
3: $J_i \leftarrow$ unsplit bins at level i
4: **for all** $i \leftarrow 0 \ldots m-1$ **do**
5: **while** $J_i \neq \emptyset$ **do**
6: $j =$ `Select`(J_i)\{Bin Selection for Refinement\}
7: $Z^* = \max(Z_i^*,$`Refine`$(Z_i^*,$ `Items`$(j)))$
8: **end while**
9: **end for**

The four major steps in $MSKP$ are detailed below.

2.4.1 Profit Approximation - `SplitMerge`

The `SplitMerge` algorithm performs the rounding and reduction step in $MSKP$. It uses an appropriate discretization of the profit axis as its starting point and creates a hierarchy of bins by splitting and merging bins as necessary. This process is controlled by two input parameters to the algorithm, a threshold for the average error for the items in a bin, τ, and the number of levels in the bin hierarchy, m. Figure 2-2 shows the bin approximation errors for a particular initial binning scheme. Figure 2-3 shows a `Merge` operation that creates a new bin with approximation error less than the threshold by combining two adjacent bins. Figure 2-4 shows a `Split` operation that divides a bin with approximation error greater than the threshold to create two bins with approximation errors less than the threshold. Algorithm 3 outlines the `SplitMerge` procedure.

The split denotes the subdivision of a bin with average approximation error greater

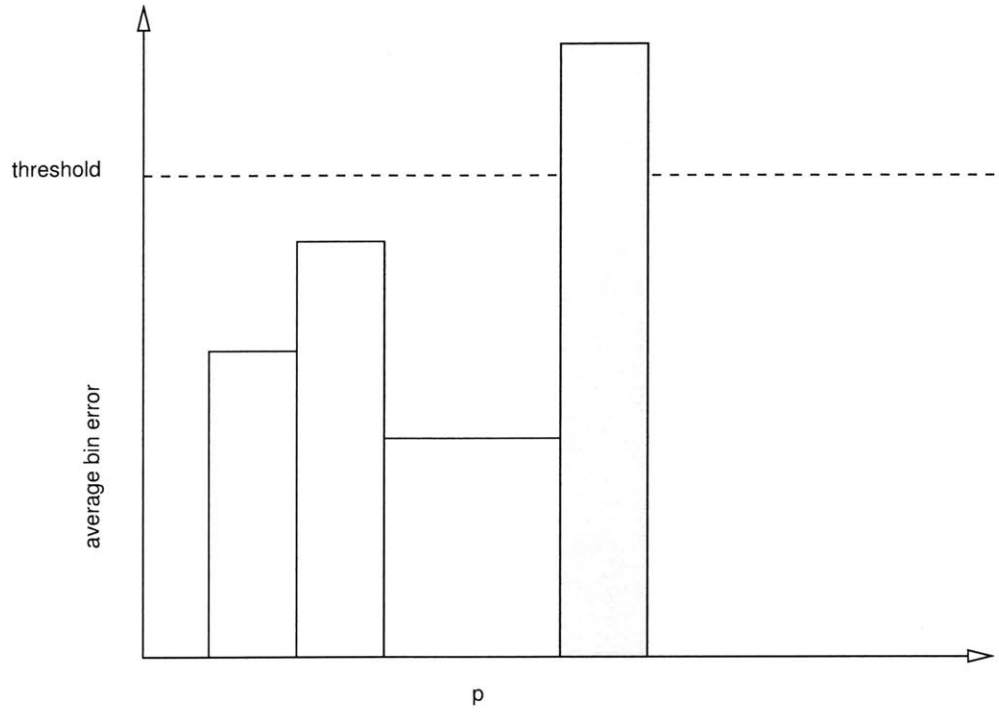


Figure 2-2: Initial Binning

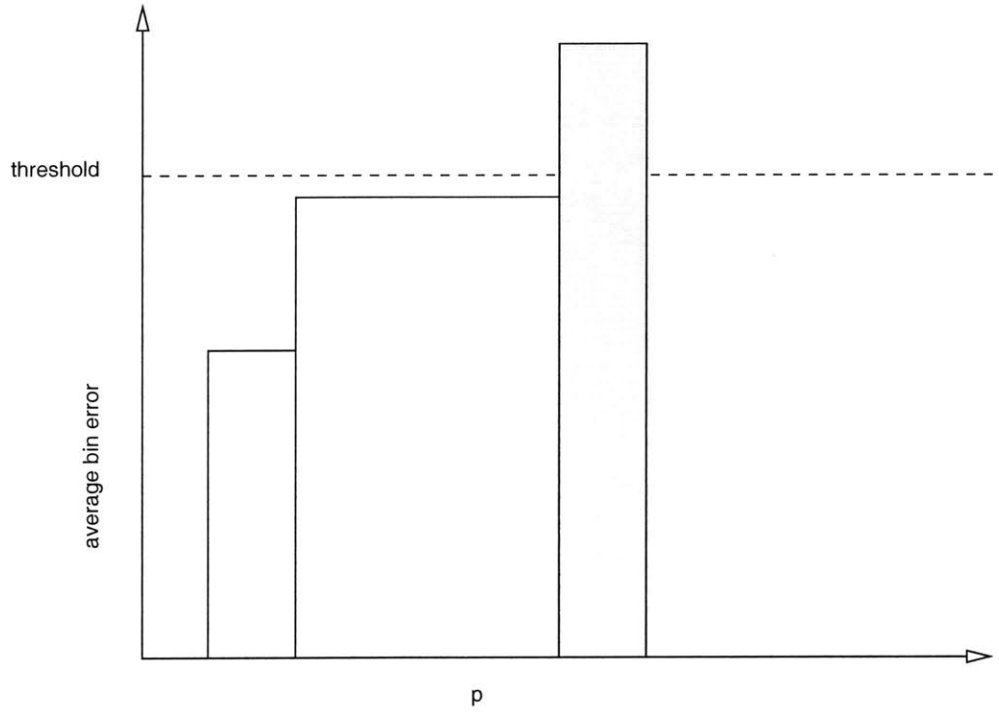


Figure 2-3: `Merge` Operation

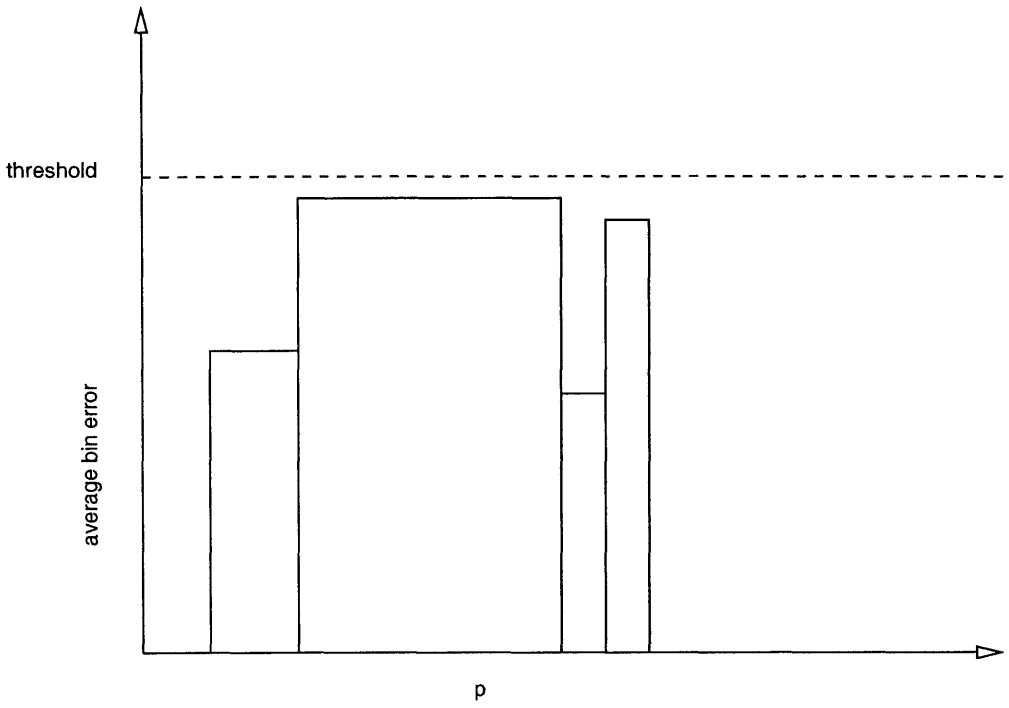


Figure 2-4: `Split` Operation

than τ. Adjacent bins are merged if the average approximation error in the combined bin is less than the threshold. The maximum number of adjacent bins that can be merged is $\leq 2^m$. This number is denoted as M in the algorithm.

The arithmetic and geometric binning schemes presented earlier discretize the profit range without taking into consideration the distribution of item profits. The `SplitMerge` algorithm is adaptive in the sense that it adjusts the bin sizes according to the item profit histogram. The output of the algorithm is the new set of bin profit values, also referred to as the coarse approximation of the problem data.

Algorithm 3 The `SplitMerge` Adaptive Binning Algorithm

Inputs
P vector of profit values
p_i profit value for the i^{th} bin in binning scheme S
p_L large profit threshold, if $p \geq p_L$, p is considered a large profit
m a parameter which denotes the maximum levels of split/merge
M the maximum number of mergeable adjacent bins $M \leq 2^m$
τ a parameter which denotes an average bin error cut off threshold

Outputs
p'_i new bin profit values

Initialize
$\mathcal{L} \leftarrow$ set of large profit bins
$\varepsilon_i \leftarrow$ average error per bin
$L \subset \mathcal{L}$ with $\varepsilon_i \leq \tau$
$S \subset \mathcal{L}$ with $\varepsilon_i > \tau$
$p'_i \leftarrow p_i$
Split

for all $i \in S$ **do**
  $j \leftarrow 1$
  while $j \leq m$ & $\varepsilon_i > \tau$ **do**
    $\{p'\} \leftarrow \frac{p_i}{2^j}$ {Split bin i into 2^j bins}
    $j \leftarrow j + 1$
  end while
end for

Merge

while $L \neq \emptyset$ **do**
  $i \leftarrow i \in L \mid \varepsilon_i$ is minimum
  $j \leftarrow j \in \{i - M, \ldots, i - 1\} \mid$ bin j is non-empty and j is maximum
  $\Delta\varepsilon_j \leftarrow$ change in ε_j if bin i is merged into j
  if $\Delta\varepsilon_j + \varepsilon_j \leq \tau$ **then**
    merge bin j into bin i
    drop p'_i from p'
    $\varepsilon_j \leftarrow \varepsilon_j + \Delta\varepsilon_j$
    mark bin i as merged
  else
    mark bin i as cannot be merged
  end if
end while

2.4.2 The Coarse Solution - `Solve`

The procedure `Solve` takes the output of `SplitMerge` and creates a coarse level solution to the problem using a pseudo-polynomial dynamic program approach.

Partition the set of items into two subsets, a set of large profit items $\mathcal{L} = \{i \ : \ p_i > \epsilon Z^H\}$ and a set of small profit items $\mathcal{S} = \{i \ : \ p_i \leq \epsilon Z^H\}$. In any feasible solution, the number of large items $N_L = \min\{k, \lfloor \frac{2}{\epsilon} \rfloor\}$, since $Z^* \leq 2Z^H$.

Following [3], Let $U \subseteq \mathcal{L}$ such that the total weight of all items in U, $w(U) \leq c$ and the cardinality of U, $|U| \leq k$. Let $p(U)$ be the total profit of all the items in U. U has the additional property that $w(U)$ is minimum among all such sets with the same profit sum and cardinality. For each such U compute a set $T \subseteq \mathcal{S}$ such that $|T| + |U| \leq k$ and $p(T)$ is maximum. The combination of U and T which has the highest profit sum is our coarse solution. We use a dynamic programming recursion to compute U.

Define set V as

$$V = \{p(U) \ : \ U \subseteq \mathcal{L} \text{ and } w(U) \leq c \text{ and} |U| \leq k\} \tag{2.2}$$

Let $\alpha = |\mathcal{L}|$ and $\beta = |V|$.

For $i = 1, \ldots, \alpha$, $a \in V$, $l = 1, \ldots, N_L$ let

$$g_i(a,l) = \min \sum_{j=1}^{i} w_j x_j \ : \ \begin{cases} \sum_{j=1}^{i} p_j x_j & = a \\ \sum_{j=1}^{i} x_j & = l \\ x_j & \in \{0,1\} \end{cases} \tag{2.3}$$

Let $g_0(0,0) = 0$ and $g_0(a,l) = +\infty$. We can then compute g_i using a recursion

$$g_i(a,l) = \min \begin{cases} g_{i-1}(a,l) \\ g_{i-1}(a-p_i, l-1) + w_i \quad \text{if } l > 0 \text{ and } a \geq p_i \end{cases}$$

2.4.3 Bin Selection for Refinement - `Select`

We can divide the set of items into two subsets, one with the optimal solution sensitive to changes in the item profits and one insensitive to the changes in item profits. We can use coarse bins for the subset that does not influence the optimum and fine bins for the subset that does. However, computing this optimal binning is as hard as solving the optimization problem itself. The hierarchical binning scheme shown above provides us with a framework to approach this ideal binning iteratively.

Let $p_{i,l}$ be the i^{th} item's profit at level l in the hierarchy of bins. Let x_i be a binary variable which indicates whether or not item i is part of the optimal solution.

We have,

$$\sum_{1}^{N} x_i p_{i,0} \leq \sum_{1}^{N} x_i p_{i,l} \leq \sum_{1}^{N} x_i p_{i,m-1}$$

Assuming that we know the optimal binning beforehand, let p_i^* be the i^{th} item's profit in the optimal binning scheme and x_i^* be the corresponding binary selection variable. $\{p_i^*\}$ will have components at various levels $l = [0, \ldots, m-1]$ and the least number of distinct values such that $\sum_1^N x_i^* p_i^* = \sum_i^N x_i^* p_{i,m-1}$. Geometrically, we can look at the $p_{i,l}$ values progressively approximating the actual profit values. However, not all the profit values are equally important; some influence the global optimum more than the others. Those components of $\{p_{i,l}\}$ to which the optimal value is not sensitive remain at the coarser levels. The following heuristic picks out the profit bins to refine. Algorithm 4 shows the profit density heuristic.

The profit density λ of an item is defined as

$$\lambda = \frac{p}{w}$$

Algorithm 4 Average Profit Density Heuristic in Select

1: $J_i \leftarrow$ unsplit bins at level i

2: $\lambda_i^k \leftarrow$ average λ of bin k at level i

3: bin to be split $k \leftarrow \arg\max_k \{\lambda_i^k - \lambda_{i-1}^k\}$

Geometrically, we are reclaiming that component of the item profit vector that has a higher chance of influencing the optimal profit while causing a considerable change in the profit vector direction. Depending on the solution accuracy desired and the time budget available, the procedure `Select` heuristically selects a coarse bin to split.

2.4.4 Solution Refinement - `Refine`

The procedure `Refine` improves the solution using the newly split bins. It is useful here to analyze the sensitivity of the optimum to perturbations of a single item's profit. The analysis is similar to that of Hifi et al. [7]. Let $p_i \rightarrow p_i + \Delta$, $\Delta \geq 0$ for any split step. Let Z be the optimum solution to the problem before the change in p_i, let X be the set of indices which are part of the solution. If $i \in X$, as Δ is positive, the optimal profit can only increase, hence i remains part of the solution.

If $i \notin X$ Let Z_i be the optimal solution to the problem under the stipulation that $x_i = 1$. By definition, we have $Z_i \leq Z$. Now if p_i is perturbed to $p_i + \Delta$, the optimal solution vector X does not change iff $Z \geq Z_i + \Delta$, or when $\Delta \in [0, Z - Z_i]$. In [7], this interval is approximated using a LP upper bound for Z, Z_i. We however have the results from the dynamic program at our disposal. Z_i is given by the combination of U and T with $x_i = 1$ and maximum profit sum. This process can be done in constant time using a lookup table for the function space of $g_i(a, l)$.

Let B_i be the set of items in bin i, now split into two bins B_{ia} and B_{ib} with profits p_i and $p_i + \Delta$ respectively. Let

$$B_{ia}^{old} = B_i \cap B_{ia}$$

$$B_{ia}^{new} = B_{ia} \setminus B_{ia}^{old}$$

$$B_{ib}^{old} = B_i \cap B_{ib}$$

$$B_{ib}^{new} = B_{ib} \setminus B_{ib}^{old}$$

Let U^* and T^* be the optimal item sets from the dynamic program. We need two sets

of feasible solutions to augment the set U: feasible solutions containing only items from B_{ib} and feasible solutions containing items from B_{ib} and items from other bins. For each bin, if the item with the lowest weight cannot be part of the optimum, then the rest of the items need not be considered.

Evaluate the set of feasible solutions containing $j = 1, \ldots, n_{ib}$ elements from B_{ib}. This can be done by identifying existing feasible solutions containing j items from B_{ib}^{old} and updating their profits by $j\Delta$. If any such solution does not exist, it has to be computed. Update the optimal values as needed. In the worst case, where B_{ib}^{old} is $\emptyset$, solutions containing $j = 1, \ldots, n_{ib}$ items from B_{ib} can be evaluated by considering the corresponding number of items from B_{ia} with profits increased by Δ and weights increased proportionally. In this case, every item in B_{ib} will be heavier than every item in B_{ia} and the number of items in B_{ib} will not be greater than the number of items in B_{ia}.

2.5 Summary

I developed a multiscale approximation algorithm for kKP that consists of a hierarchical binning scheme for item profits that generates a multiscale representation of the problem (`SplitMerge`), a coarse solution to the problem using a dynamic programming recursion (`Solve`) and a refinement step that iteratively increases the accuracy of the solution (`Select` and `Refine`). In the next chapter, I will analyze the runtime complexity of this algorithm and show that it improves upon the current best theoretical results.

3

Complexity Analysis

This chapter analyzes the run-time complexity of the $MSKP$ algorithm described in Chapter 2. I show that the $MSKP$ algorithm, with a hybrid geometric and arithmetic rounding scheme as the starting point, improves the run-time complexity of the previous best known algorithm by a factor of $O(z)$ where $z = \min\{k, \frac{1}{\epsilon}\}$ with the same performance ratio.

3.1 Definitions and Assumptions

We expect the optimum value given by $MSKP$ to be within $1-\epsilon$ of the true optimum. I will show that the $MSKP$ algorithm is polynomial in $\frac{1}{\epsilon}$ with a performance ratio of ϵ.

We are interested in solutions where the approximation error is less than or equal

to 50%.

Definition 1. $\epsilon \in \left(0, \frac{1}{2}\right]$

Bin i contains items with profit $p \in [p_i, p_{i+1})$. Every item in this bin has its profit value approximated by p_i which is called the bin profit. Choosing geometric rounding initially implies that

$$p_i = (1 - \epsilon)p_{i+1} \tag{3.1}$$

Definition 2. *Let $f_P(p)$ be the item profit density function. If $f_P(p)$ is positive and continuous, the average relative approximation error ε_i of bin i is defined as follows*

$$\varepsilon_i = \frac{\int_{p_i}^{p_{i+1}-\delta} \frac{p - p_i}{p} f_P(p)dp}{\int_{p_i}^{p_{i+1}-\delta} f_P(p)dp} \tag{3.2}$$

If $f_P(p)$ is discrete and positive, the average relative approximation error ε_i for bin i with p varying discretely from p_i to $p_{i+1} - \delta$ is defined as

$$\varepsilon_i = \frac{\sum_{p_i}^{p_{i+1}-\delta} \frac{p - p_i}{p} f_P(p)}{\sum_{p_i}^{p_{i+1}-\delta} f_P(p)} \tag{3.3}$$

where $\delta \to 0$ is a small positive constant.

3.2 Approximation Error Within a Bin

Lemma 1. *For any continuous or discrete distribution of item profits within a bin i, where the bin profits increase geometrically, the average relative approximation error $\varepsilon_i \leq \epsilon$*

Proof.

Consider the case where $f_P(p)$ is continuous and positive within a bin. $f_P(p)$ can be discontinuous at the bin boundaries, and this generalizes the case where $f_P(p)$ is continuous over the entire range of profit values.

3.2.1 Continuous Positive Profit Distribution

The average relative approximation error for bin i is given by

$$\varepsilon_i = \frac{\int_{p_i}^{p_{i+1}-\delta} \frac{p - p_i}{p} f_P(p)dp}{\int_{p_i}^{p_{i+1}-\delta} f_P(p)dp} \tag{3.4}$$

The maximum value that the profit p can have within bin i is $p_{i+1} - \delta$ where δ is vanishingly small. Using this we can rewrite Equation 3.4 as

$$\varepsilon_i \leq \frac{\int_{p_i}^{p_{i+1}-\delta} \frac{p_{i+1} - p_i - \delta}{p_{i+1} - \delta} f_P(p)dp}{\int_{p_i}^{p_{i+1}-\delta} f_P(p)dp} \tag{3.5}$$

Taking the limit with $\delta \rightarrow 0$ gives us

$$\varepsilon_i \leq \frac{p_{i+1} - p_i}{p_{i+1}} \frac{\int_{p_i}^{p_{i+1}} f_P(p)dp}{\int_{p_i}^{p_{i+1}} f_P(p)dp} \tag{3.6}$$

or

$$\varepsilon_i \leq \epsilon \tag{3.7}$$

The result also holds for the more general assumption that $f_P(p)$ is an integrable function within a bin. The assumption that $f_P(p)$ is continuous within a bin implies the existence of the integral.

3.2.2 Discrete Positive Profit Distribution

The analysis for discrete profit distributions is similar to the continuous case, with summations replacing the integrals. As before, we calculate the average relative approximation error within bin i. From Equation 3.3 we have

$$\varepsilon_i = \frac{\sum_{p_i}^{p_{i+1}-\delta} \frac{p - p_i}{p} f_P(p)}{\sum_{p_i}^{p_{i+1}-\delta} f_P(p)} \tag{3.8}$$

Again, the maximum value that p can take within bin i is $p_{i+1} - \delta$ where δ becomes vanishingly small. We can then derive an upper bound for ε_i as follows

$$\varepsilon_i \leq \frac{p_{i+1} - p_i - \delta}{p_{i+1} - \delta} \frac{\sum_{p_i}^{p_{i+1}-\delta} f_P(p)}{\sum_{p_i}^{p_{i+1}-\delta} f_P(p)} \tag{3.9}$$

Which gives us the following upper bound on ε_i

$$\varepsilon_i \leq \epsilon \tag{3.10}$$

Equations 3.7 and 3.10 prove Lemma 1.

□

3.3 Approximation Error after Merge

Lemma 2. *The average relative approximation error of a bin formed by merging M adjacent bins each with average approximation error $\varepsilon \leq \epsilon$ is bounded above by $M\epsilon$ if $M \leq \frac{3}{\epsilon} + 2$*

Proof. This proof is by induction. Our base case is when two adjacent bins are merged. We show that Lemma 2 holds true for $M = 2$. We assume that Lemma 2 is

true for $M-1$ bins and then prove that it holds true when the M^{th} bin is merged in.

For bin i let p_i be the bin profit, n_i be the number of items and ε_i be the average relative approximation error.

Now consider two adjacent bins i and $i+1$. Let bin $i+1$ be merged with bin i. All n_{i+1} items in bin $i+1$ now have a profit value of p_i.

The maximum relative error ε_{max} for an item in bin $i+1$ is given by

$$\begin{aligned}
\varepsilon_{max} &\leq \frac{p_{i+2} - p_i}{p_{i+2}} \\
&\leq \frac{p_{i+2} - p_{i+1}(1-\epsilon)}{p_{i+2}} \\
&\leq \frac{p_{i+2} - p_{i+1}}{p_{i+2}} + \frac{\epsilon p_{i+1}}{p_{i+2}} \\
&\leq 2\epsilon
\end{aligned}$$

Let $\varepsilon_{i,i+1}$ be the average relative approximation error of the merged bin

$$\varepsilon_{i,i+1} \leq \frac{n_i\varepsilon_i + n_{i+1}(2\epsilon)}{n_i + n_{i+1}}$$

As $\varepsilon_i \leq \epsilon$, we can rewrite this equation as

$$\varepsilon_{i,i+1} \leq 2\epsilon \tag{3.11}$$

Assume that merging $M-1$ bins (bin indices from i to $i+M-2$) leads to an average relative approximation error $\varepsilon_{i,i+M-2} \leq (M-1)\epsilon$. Now merge in the $(i+M-1)^{th}$ bin. The maximum approximation error of an item that was previously in bin $i+M-1$ is given by

$$\begin{aligned}
\varepsilon_{max} &\leq \frac{p_{i+M} - p_i}{p_{i+M}} \\
&\leq 1 - (1-\epsilon)^M \\
&\leq M\epsilon - \left[\frac{M(M-1)}{2}\epsilon^2\left(1 - \frac{M-2}{3}\epsilon\right)\right] \\
&\quad - \left[\frac{M(M-1)(M-2)(M-3)}{24}\epsilon^4\left(1 - \frac{M-4}{5}\epsilon\right)\right]\cdots
\end{aligned}$$

if

$$M \leq \frac{3}{\epsilon} + 2 \tag{3.12}$$

we have

$$\varepsilon_{max} \leq M\epsilon \tag{3.13}$$

Let $\varepsilon_{i,i+M-1}$ be the average relative approximation error of the M merged bins

$$\varepsilon_{i,i+M-1} \leq \frac{n_{i,i+M-2}\varepsilon_{i,i+M-2} + n_{i+M-1}(M\epsilon)}{n_{i,i+M-2} + n_{i+M-1}}$$

Using equation 3.13 and lemma 3 we have

$$\varepsilon_{i,i+M-1} \leq (M-1)\epsilon + \epsilon\frac{n_{i+M-1}}{n_{i,i+M-2} + n_{i+M-1}} \tag{3.14}$$

Hence we have

$$\varepsilon_{i,i+M-1} \leq M\epsilon \tag{3.15}$$

Using Equation 3.11 as the starting point we can see by induction that this proves Lemma 2 as long as the condition on M given by Equation 3.12 holds true. □

3.4 Complexity Analysis

3.4.1 Complexity of the Coarse Solution and Refinement

The coarse solution is computed using the dynamic program recursion (`Solve`) described in Chapter 2.

The `Solve` procedure 2.3 has been shown [3] to have a complexity of $O(\alpha\beta N_L)$. Computing T is linear in the size of $\mathcal{S}$, N_S. The overall time complexity of the coarse solution is given by $O(N + \alpha\beta N_L + \beta N_L N_S)$ where N_L is the number of large profit items in a solution, N_S is the number of small profit items, α is the number of large profit bins, β is the number of elements in the set V. Recall from equation 2.2 that

$$V = \{p(U) \ : \ U \subseteq \mathcal{L} \text{ and } w(U) \leq c \text{ and} |U| \leq k\}$$

V is the set of distinct profit values that can be attained using feasible combinations of the large profit items.

To avoid exponential complexity for the dynamic program, we make the set of possible solution profits countable by approximating each geometric bin profit with the nearest arithmetic bin of size $\frac{\epsilon Z^H}{L}$. Here L is the largest number of large profit items that can be part of the solution. This happens when only items from the smallest large bin are selected for the solution. L is of $O(\frac{1}{\epsilon})$. Merging bins reduces the number of large profit bins and the size of the set of possible solution profit values by M. Let $z = \min\{k, \frac{1}{\epsilon}\}$.

Then we have the following: α is $O(\frac{z}{M\epsilon})$, β is $O(\frac{z}{M\epsilon})$, N_L is $O(\frac{1}{\epsilon})$ and N_S is $O(\frac{k}{\epsilon})$. The pre-processing steps for creating the hierarchical bins are $O(N)$. From equation 3.12, M is $O(\frac{1}{\epsilon})$.

Hence the complexity of computing the coarse solution is given by

$$O(N + \frac{kz}{\epsilon^2}) \tag{3.16}$$

The maximum number of bins that can be refined is given by $O(\frac{1}{\epsilon^2})$. Each refinement step takes $z = \min\left(k, O(\frac{1}{\epsilon})\right)$ time. Hence the complexity of the refinement process is given by

$$O(\frac{z}{\epsilon^2}) \tag{3.17}$$

3.4.2 Complexity of the Algorithm

Theorem 1. *For any continuous or discrete distribution of item profits, there exists a fully-polynomial time approximation scheme for the kKP that runs in $O(N + \frac{kz}{\epsilon^2})$ with a performance ratio of ϵ.*

Proof. Once every bin is refined to its final state, the maximum approximation error for any item is $O(\epsilon)$ as the binning is the same as the hybrid binning scheme. The

overall complexity of the algorithm is then given by $O(N + \frac{kz}{\epsilon^2})$. □

3.5 Extensions to Other Problems

3.5.1 An $FPTAS$ for BKP

A straightforward extension of $MSKP$ is to apply the algorithm to an ordinary Binary Knapsack Problem (BKP). A wide variety of $FPTAS$ are known for BKP. Table 3.1 summarizes currently known algorithms with advanced rounding/reduction schemes.

Author	Complexity
Ibarra, Kim [8]	$O(N \ln N + \frac{1}{\epsilon^2} \min\{N, \frac{1}{\epsilon^2} \ln(\frac{1}{\epsilon})\})$
Lawler [15]	$O(N \ln \frac{1}{\epsilon} + \frac{1}{\epsilon^4})$
Magazine, Oguz [16]	$O(\frac{N^2 \ln N}{\epsilon})$
Kellerer, Pferschy [11, 10]	$O(N \min\{\ln N, \ln \frac{1}{\epsilon}\} + \frac{1}{\epsilon^2} \ln \frac{1}{\epsilon} \min\{N, \frac{1}{\epsilon} \ln \frac{1}{\epsilon}\})$
Krishnan	$O(N + \frac{N}{\epsilon^3})$

Table 3.1: $FPTAS$ for BKP

kKP assumes that N dominates k and $\frac{1}{\epsilon}$. We can extend the $FPTAS$ shown above for the kKP to solve a binary knapsack problem (BKP) by setting $k = N$. The complexity of this algorithm is then $O(N + \frac{N}{\epsilon^3})$.

3.5.2 Cost of Re-optimization

Consider an online variation to the kKP where we first solve the problem for a given initial set of items and then as new items come in, we recompute the optimal solution.

We assume that every bin already contains the maximum number of items that it can hold.

In each bin, all the items are assigned the same profit value and are ordered according to weight. The new item introduced into the bin can be

1. heavier than any existing item in the bin
2. lighter than some r existing items in the bin

For case 1, the optimal solution does not change as the bins are saturated. For case 2, the optimal solution might change as the new item will replace the item next to it in weight in all interesting solutions that it appears in. In effect, this trickles down the bin. The total computational effort is then proportional to the number of items that are displaced ($O(r) \leq O(\min(k, \frac{1}{\epsilon}))$).

3.6 Conclusions

In this chapter I considered discrete or continuous item profit density functions and showed that the runtime complexity of the $MSKP$ algorithm is $O(N + \frac{kz}{\epsilon^2})$. In Appendix B, I look at specific continuous distributions for the item profit, namely, linear, power and exponential and give alternate derivations for the runtime complexity of $MSKP$ in those cases. The next chapter presents the results of an empirical study comparing the run times and solution quality of $MSKP$ against a commercial mixed integer program solver.

4

Computational Results

In this chapter, I explore the computational performance of the multiscale approximation algorithm ($MSKP$) and the industry standard CPLEX 9.0 [9] software package. I show that for a variety of cases, the running time performance of $MSKP$ matches or does better than that of CPLEX, though with some loss in solution quality.

4.1 Problem Generation

A variety of Knapsack problems were generated using the advanced $0-1$ Knapsack problem generator referenced in [23]. These problem instances were modified to add a cardinality constraint and then transformed into formats readable by CPLEX and $MSKP$ using opbdp [2]. All experiments were run on a computer with a 1.86GHz Intel Pentium-M processor with 2GB of memory. Following guidelines in the literature

(for example, Martello and Toth [17]) we construct three classes of problems.

4.1.1 Uncorrelated Instances

Here there is no correlation between the profit and weight of a given item. The profits and weights are sampled from uniform distributions, $p_j \backsim U(1, P)$, $w_j \backsim U(1, W)$. Figure 4-1 shows the scatter plot for such an instance.

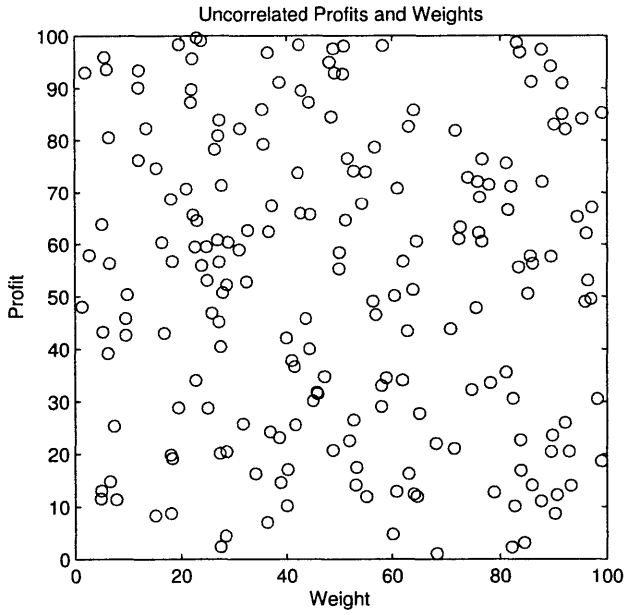


Figure 4-1: Uncorrelated Profits and Weights

4.1.2 Weakly Correlated Instances

Here the item profit is weakly correlated with the item weight. If the item weights are sampled from a uniform distribution, $w_j \backsim U(1, W)$, the item profits are within a band δ of the item weights. $p_j = w_j \pm U(1, \delta)$. Figure 4-2 shows the scatter plot for such an instance.

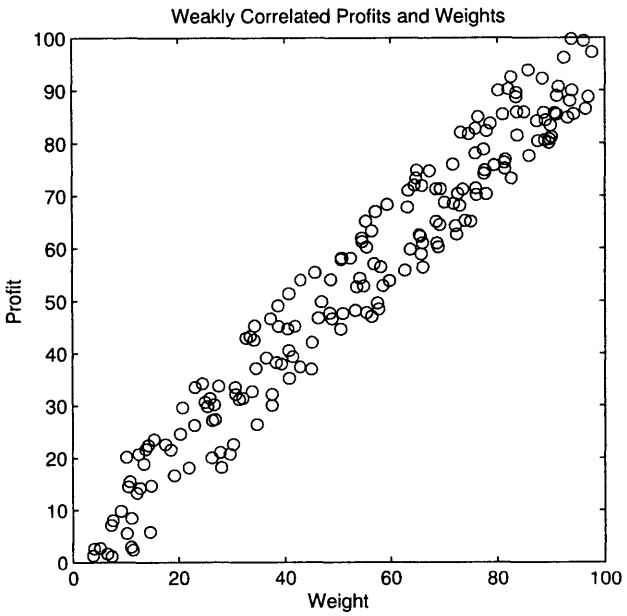


Figure 4-2: Weakly Correlated Profits and Weights

4.1.3 Strongly Correlated Instances

Here the item weights are sampled from a uniform distribution and the item profit is a linear function of the item weight. Figure 4-3 shows a scatter plot of such an instance. For the BKP, the strongly correlated case tends to be the hardest to solve in practice.

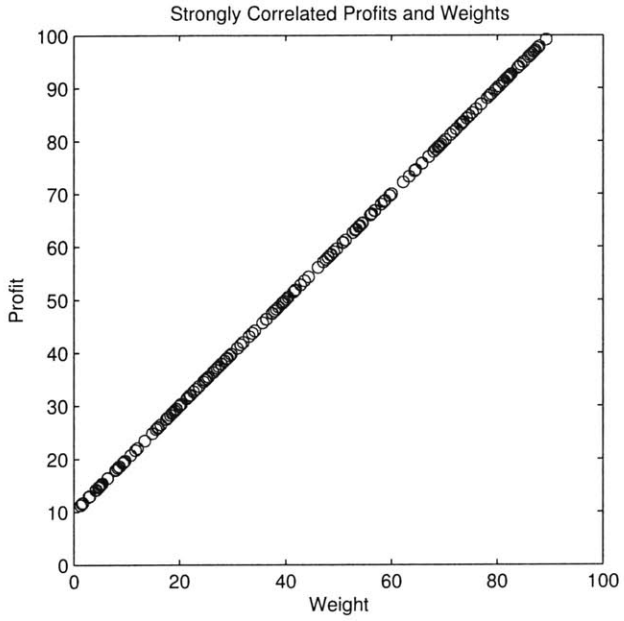


Figure 4-3: Strongly Correlated Profits and Weights

4.2 Comparison of Hybrid Binning and $MSKP$

$MSKP$ and Hybrid Binning (HB) share the same core dynamic programming solution. Computational experiments show that HB becomes intractable when the problem size increases and the acceptable error goes down. For most of the problem instances considered in this study, the running time of HB was in the order of 10^5ms or more. Table 4.1 compares the average running times of HB, $MSKP$ and CPLEX over the three broad categories of problems considered.

Problem Instance	**Average Running Times (ms)**		
	CPLEX	$MSKP$	HB
Uncorrelated	192.50	20.00	$\geq 10^5$
Weakly Correlated	192.88	57.50	$\geq 10^5$
Strongly Correlated	175.00	167.50	$\geq 10^5$

Table 4.1: Hybrid Binning vs $MSKP$ vs CPLEX

4.3 Numerical Results for Varying Capacity

For each problem class, 8 random instances were generated and tested 5 times each with an allowable error setting $\epsilon = 4\%$. The optimal solution and running time for $MSKP$ implemented in C++ were compared to results from CPLEX. All problems generated have $N = 1000$ items following the guidelines for generating reasonably large problems in previous experiments [5, 22].

4.3.1 Uncorrelated Instance

Table 4.2 shows the computational results for datasets where the profits and weights are uncorrelated. The problem was formulated with $N = 1000$ and $k = 30$. For each of the 8 instances a different capacity constraint c was chosen. The $MSKP$ algorithm had an allowable error setting $\epsilon = 4\%$ and up to 6 levels of decomposition. $MSKP$ achieved a better than 96% solution with an average speedup of 9.63 over the CPLEX solution. Figure 4-4 shows the running time comparison between CPLEX

and $MSKP$. Figure 4-5 shows the optimal solution given by both algorithms and Figure 4-6 shows the acceptable maximum and actual solution errors for $MSKP$.

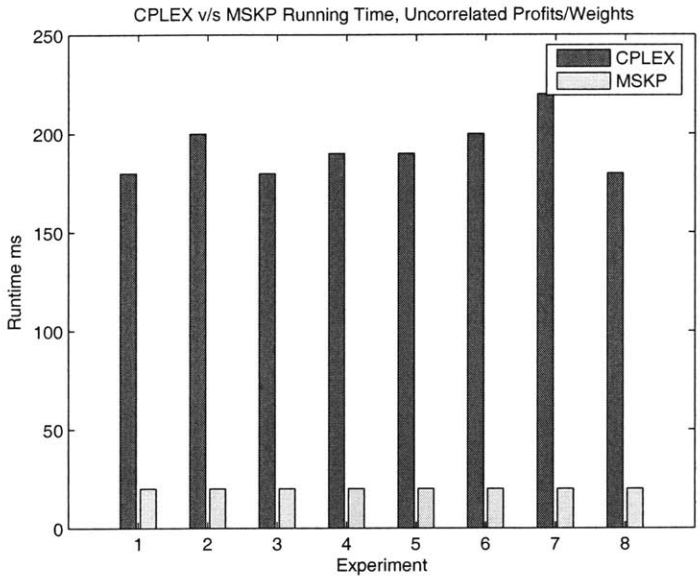


Figure 4-4: Uncorrelated Problem Running Times

Exp	Capacity	CPLEX		MSKP		Error	Speed Up
		OPT	t (ms)	OPT	t (ms)		
1	40	1447	180	1419	20	0.019	9.00
2	43	1507	200	1494	20	0.009	10.00
3	49	1627	180	1608	20	0.012	9.00
4	51	1661	190	1642	20	0.011	9.50
5	55	1734	190	1728	20	0.003	9.50
6	69	1955	200	1955	20	0.000	10.00
7	73	2014	220	2001	20	0.006	11.00
8	80	2118	180	2118	20	0.000	9.00

Table 4.2: Uncorrelated Instances with 4% Acceptable Error

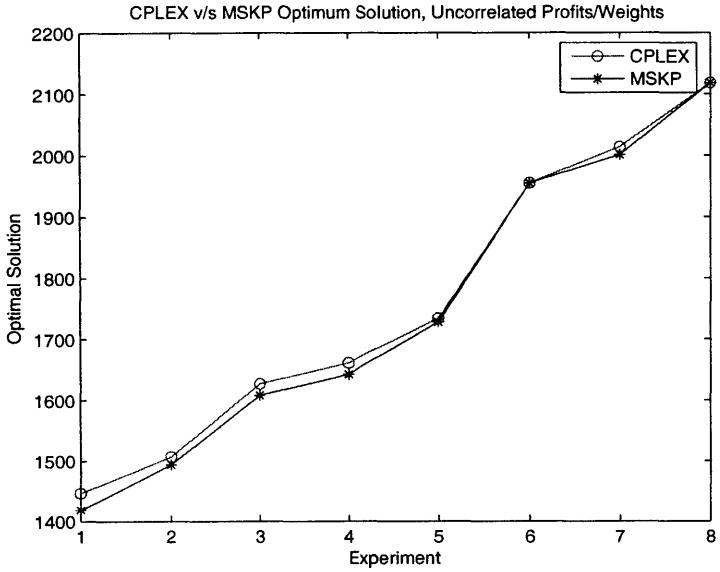


Figure 4-5: Uncorrelated Problem Optimal Solutions

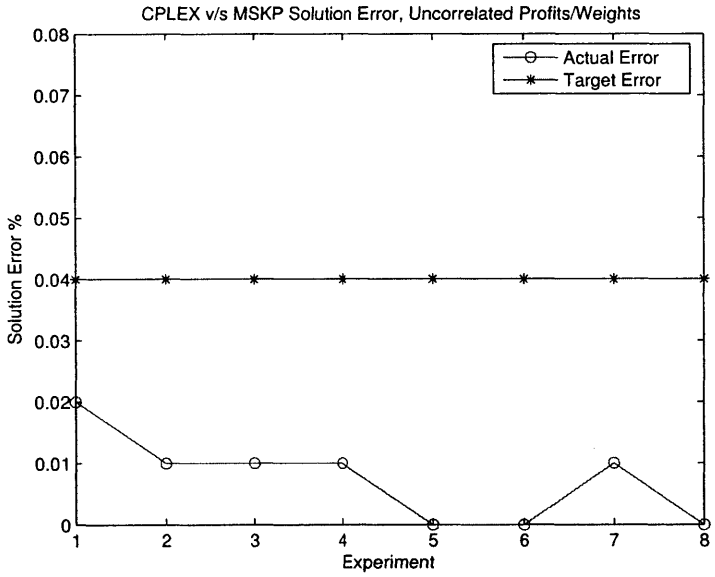


Figure 4-6: Uncorrelated Problem Solution Error

4.3.2 Weakly Correlated Instance

Table 4.3 shows the numerical results for problem instances where item profits and weights were weakly correlated. Again, the problem size was $N = 1000$ and $k = 30$. Each of the eight experiments had a different capacity setting. The $MSKP$ algorithm had up to 6 levels of decomposition and an acceptable error of 4%. Figure 4-7 shows the running time comparison between CPLEX and $MSKP$. Figure 4-8 and Figure 4-9 show the optimal solution and solution error for CPLEX and $MSKP$. We can see that $MSKP$ achieved a better than 96% solution with an average speedup of 3.39 over the CPLEX solution.

Exp	Capacity	CPLEX		MSKP		Error	Speed Up
		OPT	t (ms)	OPT	t (ms)		
9	100	244	195	237	60	0.029	3.25
10	112	266	188	257	60	0.034	3.13
11	126	292	200	281	50	0.038	4.00
12	152	340	160	339	60	0.003	2.67
13	169	369	210	364	50	0.014	4.20
14	181	390	200	383	60	0.018	3.33
15	193	410	190	404	60	0.015	3.17
16	206	430	200	415	60	0.035	3.33

Table 4.3: Weakly Correlated Instances with 4% Acceptable Error

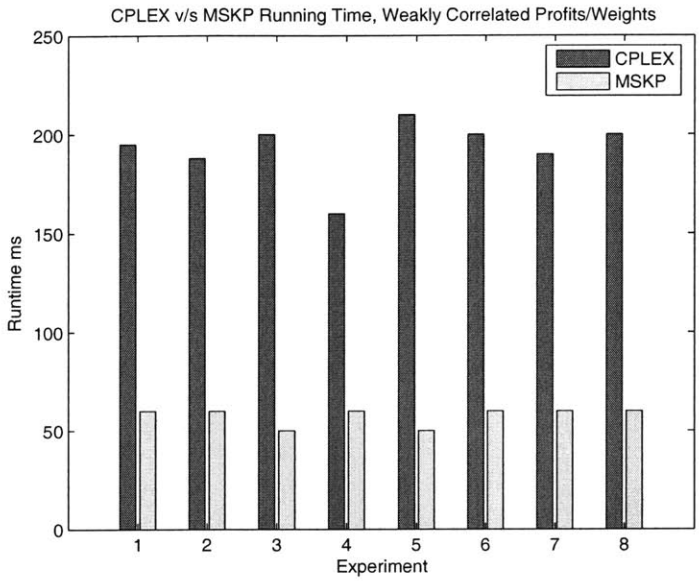


Figure 4-7: Weakly Correlated Problem Running Times

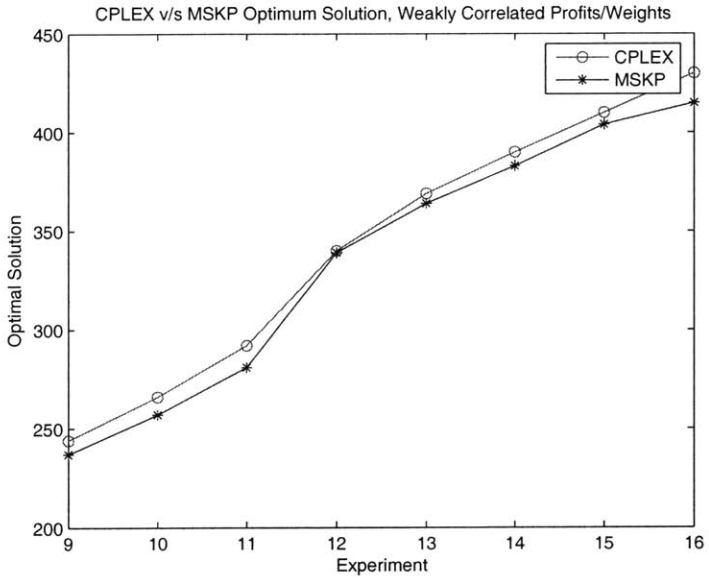


Figure 4-8: Weakly Correlated Problem Optimal Solutions

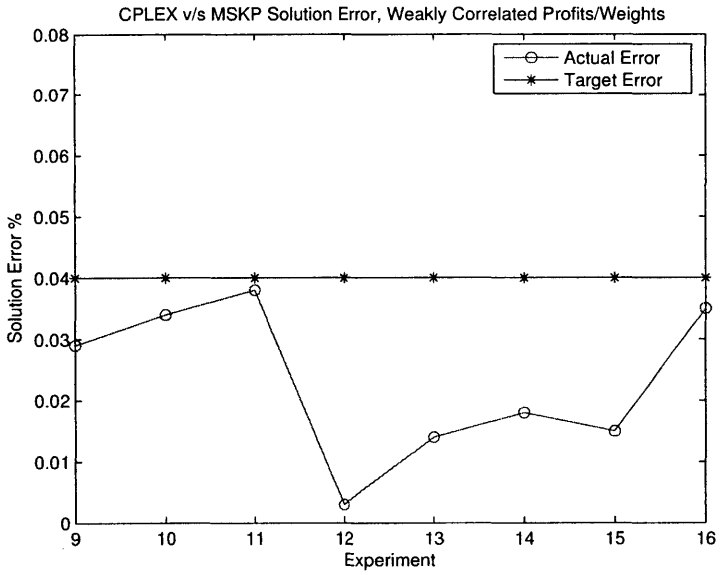


Figure 4-9: Weakly Correlated Problem Solution Error

4.3.3 Strongly Correlated Instance

Table 4.4 shows the numerical results for problem instances where item profits and weights were strongly correlated. The settings for the $MSKP$ algorithm were an acceptable error of 4% and up to 6 levels of decomposition. Figure 4-10 shows the runtime comparison between CPLEX and $MSKP$. As we can see from Figure 4-11 and Figure 4-12, the $MSKP$ solution was the optimal solution and the average runtime speedup factor was 1.11. As the capacity constraint became looser, the runtime performance of the CPLEX program started edging ahead of $MSKP$. $MSKP$ performance becomes better as the capacity constraint becomes tighter.

For the strongly correlated cases, the $MSKP$ algorithm consistently produced zero error solutions. In such cases, the high correlation between the weights and profits implies that almost no information is lost when the profits are approximated. This might be a plausible explanation for this behavior.

Exp	Capacity	CPLEX		MSKP		Error	Speed Up
		OPT	t (ms)	OPT	t (ms)		
17	100	400	180	400	120	0.000	1.50
18	111	411	180	411	130	0.000	1.38
19	125	425	180	425	120	0.000	1.50
20	143	443	170	443	160	0.000	1.06
21	167	467	150	467	170	0.000	0.88
22	188	488	180	488	180	0.000	1.00
23	215	515	180	515	210	0.000	0.86
24	251	551	180	551	250	0.000	0.72

Table 4.4: Strongly Correlated Instances with 4% Acceptable Error

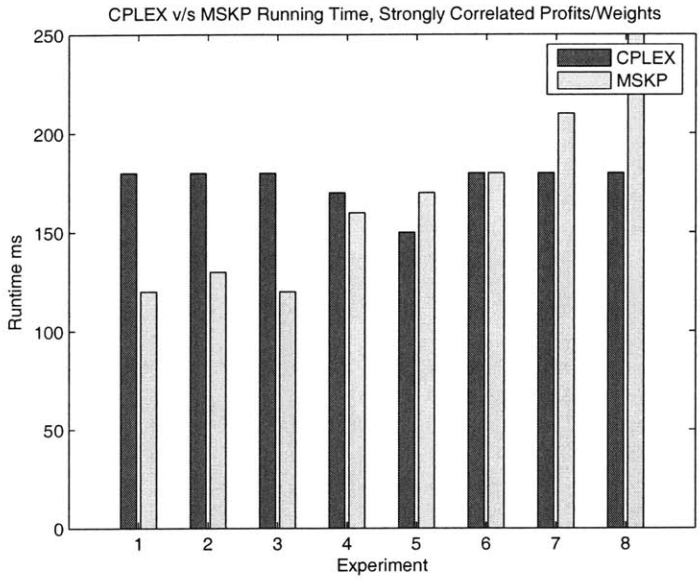


Figure 4-10: Strongly Correlated Problem Running Times

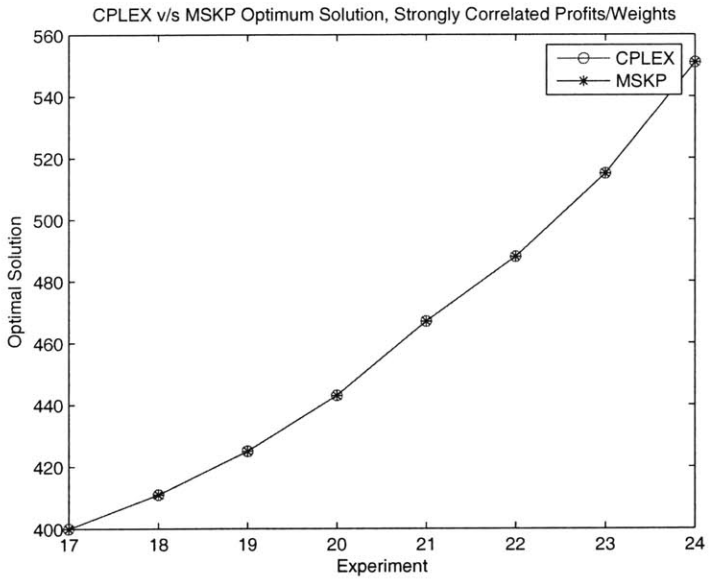


Figure 4-11: Strongly Correlated Problem Optimal Solutions

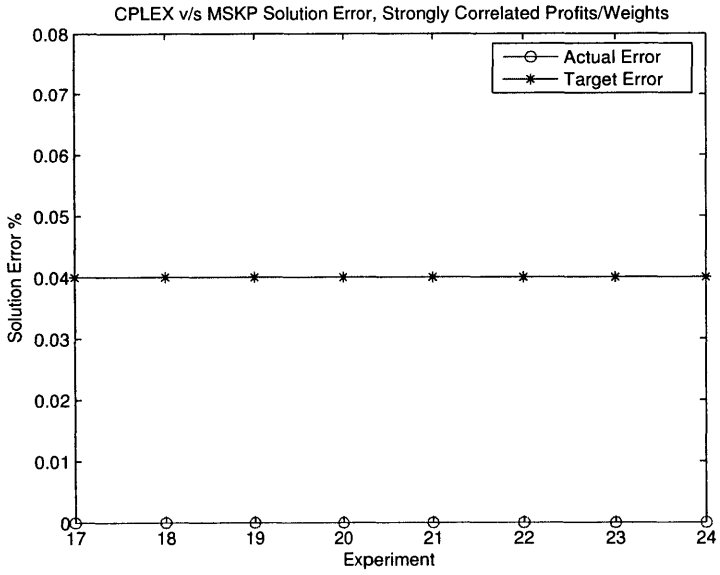


Figure 4-12: Strongly Correlated Problem Solution Error

4.4 Numerical Results for Varying k

In this experiment, we hold the capacity fixed for a given uncorrelated problem and vary the cardinality constraint k from 20 to 29. Table 4.5 summarizes the results for acceptable error set to 4%. Figure 4-13 shows running time comparisons between CPLEX and $MSKP$. On an average, $MSKP$ is around 6.5 times faster than CPLEX with an average solution error of 1.6%.

Figure 4-15 and Figure 4-14 show the optimal solution and error values for the $\epsilon = 0.04$ case. As the k value decreases, for the same capacity constraint, the number of options that a solver has to consider from a given set of items increases and this leads to the longer computation times. This behaviour is not seen in CPLEX as it uses a branch-cut strategy rather than a dynamic programming procedure in this case.

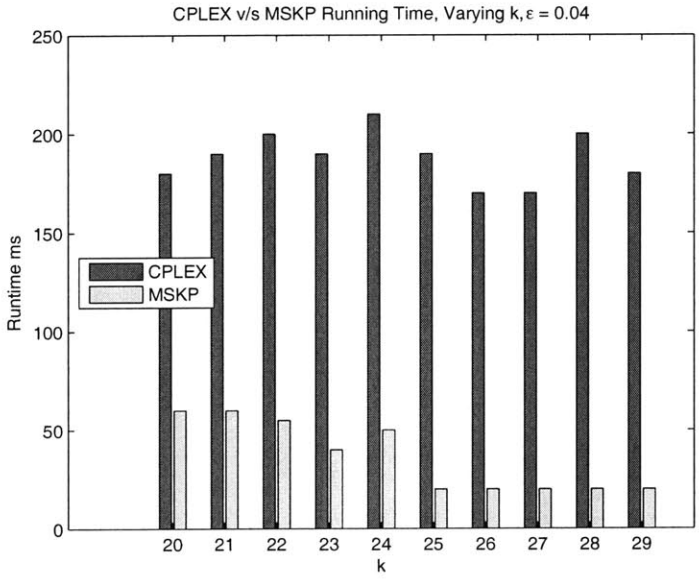


Figure 4-13: Running Times for Varying k, $\epsilon = 0.04$

ExpId	k	CPLEX Sol	CPLEX t (ms)	$MSKP$ Sol	$MSKP$ t (ms)	Solution Error	Speedup
211	29	2332	180	2290	20	0.0180	8.667
212	28	2304	200	2269	20	0.0151	9.833
213	27	2277	170	2232	20	0.0197	8.500
214	26	2233	170	2173	20	0.0268	8.500
215	25	2181	190	2139	20	0.0192	9.500
216	24	2127	210	2117	50	0.0047	4.350
217	23	2067	190	1992	40	0.0362	5.527
218	22	2001	200	1990	55	0.0054	4.083
219	21	1928	190	1915	60	0.0067	3.333
220	20	1855	180	1827	60	0.0150	2.889

Table 4.5: Results for varying k with $\epsilon = 0.04$

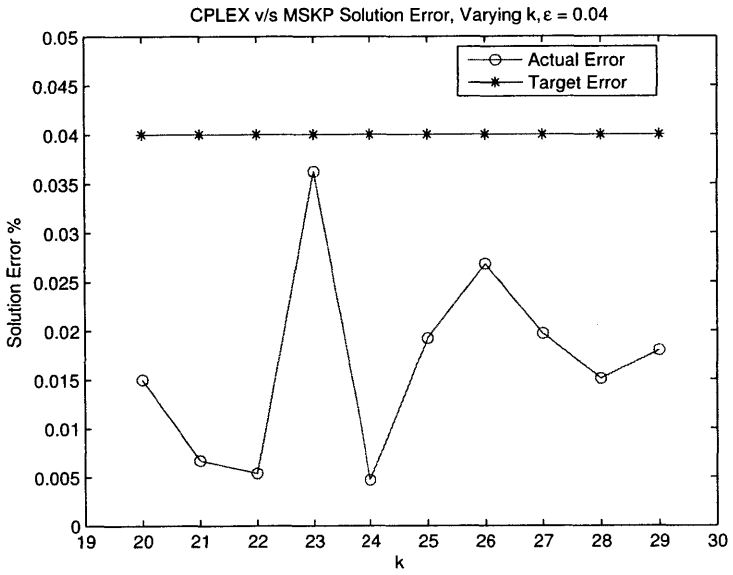


Figure 4-14: Solution Error for Varying k, $\epsilon = 0.04$

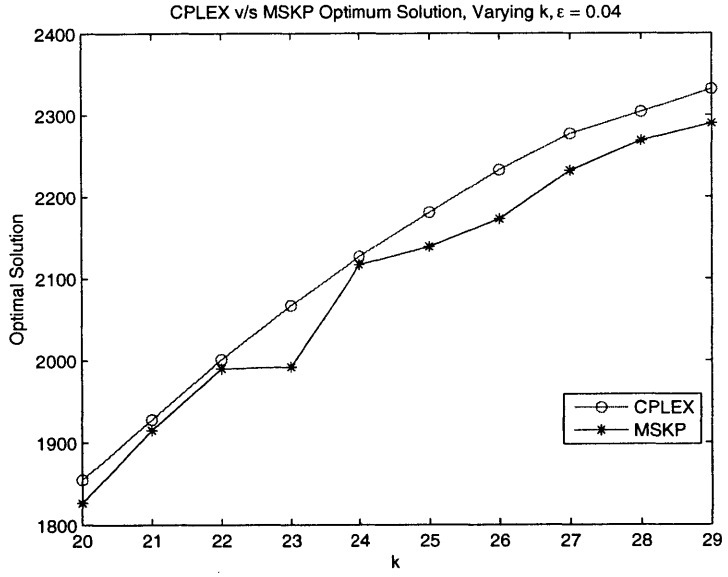


Figure 4-15: Optimal Solutions for Varying k, $\epsilon = 0.04$

4.5 Numerical Results for Varying ϵ

Here we keep the problem instance constant and vary the allowable error parameter ϵ for the $MSKP$ algorithm. The problem instance has uncorrelated profits and weights. We set $N = 1000$, capacity to 101 and $k = 30$ and up to $m = 4$ bin levels. We run 5 experiments each for 7 values of ϵ between 2% and 3.5%. The exact solution value for this problem was calculated using CPLEX to be 2354 in 180ms. Table 4.6 summarizes the results. Figure 4-16 shows the running times of $MSKP$ as ϵ changes. Figure 4-17 compares the solution quality for different acceptable error upper bounds.

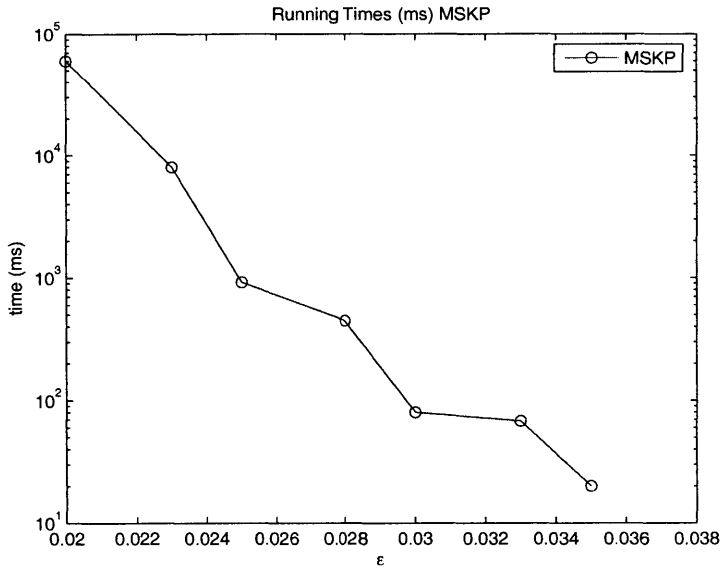


Figure 4-16: Running Times for Varying ϵ

Exp. ID	ϵ	$MSKP$(Opt)	$MSKP$(t ms)	Solution Error
1	0.020	2347	59919.4	0.0030
2	0.023	2345	7987	0.0038
3	0.025	2329	919	0.0106
4	0.028	2338	448	0.0068
5	0.030	2338	80	0.0068
6	0.033	2328	68	0.0110
7	0.035	2332	20	0.0090

Table 4.6: Running Times for Varying ϵ

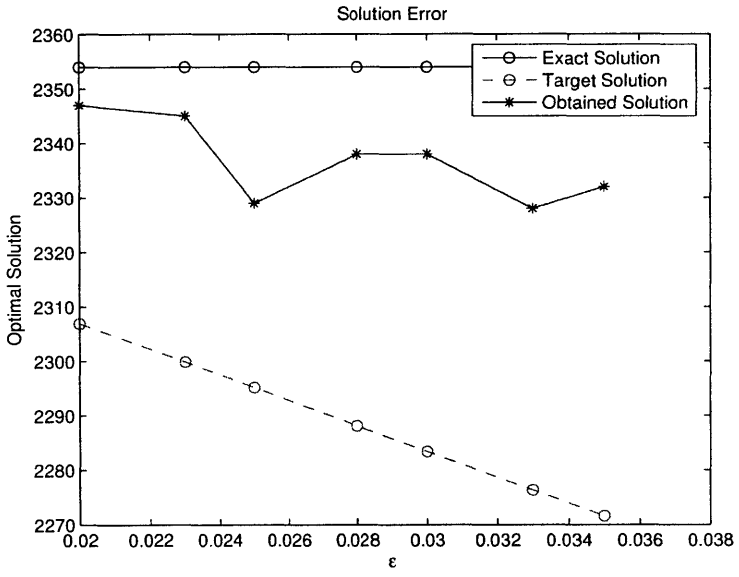


Figure 4-17: Solution Error for Varying ϵ

4.6 Summary

Table 4.7 summarizes the empirical results for problems of similar size with an acceptable error of 4%. Of the three cases considered, the uncorrelated instances had the greatest speedup. The weakly correlated cases had the highest observed error and the strongly correlated cases consistently showed zero error. In terms of running times, the strongly correlated cases were closest to CPLEX. The un-correlated problems have the widest range of profit densities while the strongly correlated problems have a single profit density (before approximation). As the variation in profit densities decreases, solving the coarse problem using dynamic programming becomes harder. The speedup results are consistent with this hypothesis. Appendix C contains additional empirical results.

Type	**Error**	**Speed Up**
Uncorrelated	0.008	9.63
Weakly Correlated	0.023	3.39
Strongly Correlated	0.000	1.11

Table 4.7: 4% Expected Error Summary

In this chapter we looked at the computational performance of the $MSKP$ algorithm introduced in the Chapter 2. For a variety of randomly generated problems we see that the running time and solution quality of $MSKP$ is comparable to that of CPLEX.

5

Conclusions

This chapter summarizes the ideas and contributions presented in this thesis and suggests some avenues for further research.

5.1 Summary

As described in Chapter 1, the aim of this thesis was to develop an approximation algorithm for the cardinality constrained knapsack problem. To this end, I developed $MSKP$, a multiscale approximation algorithm described in Chapter 2. The first step in $MSKP$ is a rounding and reduction procedure that constructs a hierarchy of bins on the profit axis. The error in the approximation increases as the bins go from fine to coarse. In the next step, a feasible solution is calculated to the problem using dynamic programming with the coarse profit bins as the input. The accuracy of this

solution is then improved by moving down the scales to the finer levels and refining the solution as needed.

Chapter 3 showed that the $MSKP$ algorithm is a fully polynomial approximation scheme and the runtime complexity is better than the previous best theoretical result (hybrid binning) for the same performance ratio. Table 5.1 shows a comparison of the runtime complexity of $MSKP$ with the two previously known algorithms.

Author	**Complexity**
Caprara et al [3]	$O(\frac{Nk^2}{\epsilon})$
Hutter, Mastrolilli [18]	$O(N + \frac{k}{\epsilon^3} \min\{k, \frac{1}{\epsilon}\})$
$MSKP$	$O(N + \frac{k}{\epsilon^2} \min\{k, \frac{1}{\epsilon}\})$

Table 5.1: $FPTAS$ for kKP

As kKP generalizes the ordinary binary knapsack problem (BKP), the algorithm also provided a fully polynomial approximation scheme for the BKP.

For a variety of problem instances, I evaluated the practical utility of the algorithm by comparing its running times to that of a commercial integer programming solver. Empirical results showed that for acceptable approximation errors, the running times of $MSKP$ was equal to or better than that of CPLEX. For example, for a problem size of a 1000 variables with uncorrelated profits and weights, $MSKP$ shows a one order of magnitude average speed up with less than 3% error as compared to CPLEX.

The general approach in developing approximation algorithms for combinatorial optimization problems often starts with rounding and reduction. A systematic process, such as the one presented in this thesis, to cast the problem data into a hierarchy defined by the approximation error may have theoretical and practical utility in solving other hard combinatorial problems. The next section presents some directions for further work.

5.2 Further Research

1. *Improving the Approximation Scheme* The arithmetic, geometric and hybrid schemes worked on the profit axis without considering the distribution of items along that axis. The `Split/Merge` scheme presented in this thesis adapts the bins to the item profit distribution by using an average approximation error within a bin as a control. It might be possible to improve the algorithm using better threshold selection schemes. The choice of $\tau = M\epsilon$ as the threshold works for all integrable distributions of item profits. However, for specific profit distributions, it might be possible to obtain similar levels of binning with lower thresholds. This will serve to increase the accuracy of the coarse solution.

   The error measure proposed in this thesis depends only upon the item profits and its distribution. Intuitively, we would stand to lose less if we increase the approximation error on those items with smaller profit densities. This may not necessarily lead to a decrease in complexity, but the empirical running times of the algorithm should improve.

2. *Explaining the Curious Behaviour of Strongly Correlated Problems* The empirical results for the strongly correlated problems in Chapter 4 and in Appendix C show that the $MSKP$ algorithm always reaches the exact solution to the problem. I hypothesize that the reason for this behaviour is that the 100% correlation between item profits and weights in this case lead to no loss of information when the profits are approximated. The change in item profits is too small to affect the optimization direction and solution location on the plane of constraints. In effect, we end up solving the original problem. An interesting direction for further research would be to see if this hypothesis is valid.

3. *Approximate Dynamic Programming for Knapsack Problems* In approximate dynamic programming (ADP), the intermediate values of the dynamic programming functional is approximated using various techniques to speed up the recursion. Demir [5] shows the empirical performance of ADP in solving knap-

sack problems and its variations. In this process, the domain over which the functional is evaluated is discretized, the functional is evaluated at the grid points and intermediate values are interpolated. A possible application of the ideas described in this thesis is the construction of an adaptive grid, which is coarse in those areas where we are not interested and finer in those areas where calculating the functional accurately is important.

4. *Other Combinatorial Problems* The multiscale approximation technique presented here might possibly be used to obtain fast heuristic solutions for other combinatorial problems. For example, in a two dimensional Euclidean traveling salesman problem, the discretization of the plane into a grid can be adapted to the distribution of locations to be visited. Another problem of practical interest comes from retail space allocation, where given limited shelf space and a limited number of shelves, the number of facings of different brands of items have to be determined such as to maximize the expected revenue. Any given brand has only one set of items on display, this adds additional cardinality constraints for each brand to be considered. While a $FPTAS$ for this problem might not be possible, a fast heuristic solution is definitely desirable.

A

List of Symbols

$\mathcal{A}$	An approximation algorithm, page 12
α	Cardinality of $\mathcal{L}$, page 26
β	Cardinality of V, page 26
BKP	The binary knapsack problem, page 11
Δ	Change in item profit when a bin is split, page 28
ϵ	Target error, performance ratio, page 12
$FPTAS$	Fully Polynomial time approximation scheme, page 12
kKP	The binary knapsack problem with cardinality constraints, page 11
λ	Profit density or the ratio of an item's profit to its weight, page 27
λ_i^k	Profit density of bin k at level i, page 27
$\mathcal{L}$	Set of large profit items, page 26
$\mathcal{S}$	Set of small profit items, page 26

$MSKP$	The multiscale approximation algorithm, page 18
Π	A maximization problem, page 12
$PTAS$	Polynomial time approximation scheme, page 12
$\Re^+$	The positive Real axis, page 11
τ	Threshold for average approximation error in a bin, page 22
ε	Average relative approximation error for bin i, page 31
ε_i	Average relative approximation error in bin i, page 24
$\varepsilon_{i,j}$	Average relative approximation error of $j-i+1$ merged bins, page 35
ε_{max}	Maximum relative approximation error for an item, page 35
a	An element of set V, page 26
B_i	Set of items in bin i, page 29
B_{ia}	Set of items in the lower profit bin created by splitting bin i, page 29
B_{ia}^{new}	Intersection of B_{ia} and B_i, page 29
B_{ia}^{old}	New items added to bin B_{ia} after bin i is split, page 29
B_{ib}	Set of items in higher profit bin created by splitting bin i, page 29
B_{ib}^{new}	Intersection of B_{ib} and B_i, page 29
B_{ib}^{old}	New items added to bin B_{ib} after bin i is split, page 29
c	The capacity of the knapsack, page 11
$f_P(p)$	Profit density distribution function, page 31
H	A heuristic algorithm, page 21
I	Index set of items in optimal solution, page 21
I_0*	Index set with $x_i = 0$, page 21
I_1	Index set with $x_i = 1$, page 21
J_i	Unsplit bins at level i, page 27
k	The cardinality constraint, maximum number of items selected in the solution, page 11
l	$l = 1, \ldots, N_L$, page 26
M	Maximum number of consecutive bins that can be merged, also 2^m, page 33
m	Number of levels of binning, page 22
N	The number of items, page 11

N_L	Number of large profit items in solution, page 26
N_S	Cardinality of set $\mathcal{S}$, page 35
n_{ib}	Number of items in bin B_{ib}, page 29
n_i	Number of items in the i^{th} bin, page 35
$OPT(\Pi)$	Optimal solution to maximization problem Π, page 12
$p^{'}$	Item profits after `Split/Merge`, page 22
p_i	The profit of the i^{th} item or bin, page 11
p_L	Small/Large profit threshold, page 24
$p_{i,l}$	Profit of item i at level l, page 27
p_{max}	Maximum item profit, page 21
T	Subset of $\mathcal{S}$ such that the total number of items in T and U is $\leq k$ and the total weight of all the items in T is less than or equal to any excess weight left over after the items in a corresponding set U fill the knapsack. Under such conditions, the total profit of the items in T is the largest possible, page 26
U	Subset of $\mathcal{L}$ such that for the total profit of all the items in U, the total weight is the smallest and also less than the capacity c of the knapsack. Also the cardinality of U is $\leq k$, page 26
V	Set of total profits of all feasible subsets U of $\mathcal{L}$, page 26
w_i	The weight of the i^{th} item, page 11
x_i	A binary variable which is 1 if item i is included in the knapsack solution, page 11
z	$\min\{k, \frac{1}{\epsilon}\}$, page 36
Z^*	Optimal solution, page 21
Z^H	Greedy heuristic solution, page 21
Z_0^*	Coarse level solution, page 22
Z_i	Optimal solution that always includes item i, page 28

B

Complexity Analysis for Selected Distributions

This chapter considers three different functional forms for the profit density function $f_P(p)$; linear, power and exponential. In each case, we prove that a specific variation of Lemma 1 holds true.

B.1 Piecewise Linear Profit Distribution

We assume that the distribution of item profits is piece-wise linear. Within a single bin we assume that this distribution can described as

$$f_P(p) = \alpha + \beta p \tag{B.1}$$

where the parameters α, β describe the behavior of the curve, p is the profit variable. We consider three possible cases for α and β.

Case 1 , $\beta = 0$, i.e., the profit distribution is constant

Case 2 , $\beta < 0$, i.e., the profit distribution has a negative slope

Case 3 , $\beta > 0$, i.e., the profit distribution has a positive slope

For each case, we compute the average relative approximation error and show that its upper bound is ϵ.

Lemma 3. *For any linear distribution of item profits within a bin i, where the bin profits increase geometrically, the average relative approximation error* $\varepsilon_i \leq \epsilon$

Proof.

B.1.1 Case 1, $\beta = 0$

Here $f_P(p)$ is a constant and is given by

$$f_P(p) = \alpha \tag{B.2}$$

where $\alpha > 0$.

ε_i is then given by

$$\varepsilon_i = \frac{\int_{p_i}^{p_{i+1}} \frac{p - p_i}{p} \alpha dp}{\int_{p_i}^{p_{i+1}} \alpha dp} \tag{B.3}$$

$$\varepsilon_i = 1 - \frac{(p_{i+1} - p_i)\alpha - \alpha p_i \ln \frac{1}{1-\epsilon}}{(p_{i+1} - p_i)\alpha} \tag{B.4}$$

$$\varepsilon_i = 1 - \frac{1-\epsilon}{\epsilon} \ln \frac{1}{1-\epsilon} \tag{B.5}$$

The Taylor series expansion for $\ln \frac{1}{1-\epsilon}$

$$\ln \frac{1}{1-\epsilon} = \epsilon + \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} + \cdots \tag{B.6}$$

gives us

$$\varepsilon_i \leq \epsilon \tag{B.7}$$

B.1.2 Case 2, $\beta < 0$

We re-write the profit density function as follows

$$f_P(p) = \alpha - \gamma p$$

where $\gamma = |\beta|$.

By definition, the density function cannot be negative, hence for every profit value p in bin i, we have the following condition

$$\alpha - \gamma p \geq 0 \tag{B.8}$$

This condition should be satisfied when p attains its maximum value within the bin. Hence we have

$$\alpha \geq \gamma p_{i+1} \tag{B.9}$$

Substituting for $f_P(p)$ in equation 3.2 gives us

$$\varepsilon_i = 1 - \frac{\int_{p_i}^{p_{i+1}} \frac{p_i}{p} (\alpha - \gamma p)\, dp}{\int_{p_i}^{p_{i+1}} (\alpha - \gamma p)\, dp}$$

To simplify the calculation, we re-write this as

$$\varepsilon_i = 1 - \frac{\mathcal{N}}{\mathcal{D}}$$

$$\mathcal{N} = \alpha p_i \ln\left(\frac{1}{1-\epsilon}\right) - \gamma p_i^2 \frac{\epsilon}{1-\epsilon}$$

$$\mathcal{D} = \alpha p_i \frac{\epsilon}{1-\epsilon} - \frac{\gamma p_i^2}{2} \frac{\epsilon\,(2-\epsilon)}{(1-\epsilon)^2}$$

Simplifying these expressions, we get

$$\frac{\mathcal{N}}{\mathcal{D}} = \frac{\frac{1-\epsilon}{\epsilon}\alpha \ln\left(\frac{1}{1-\epsilon}\right) - \gamma p_i}{\alpha - \frac{\gamma p_i (2-\epsilon)}{2(1-\epsilon)}} \tag{B.10}$$

From equation B.9 we write

$$\alpha = C\frac{\gamma p_i}{1-\epsilon}$$

where C is a positive constant ≥ 1. Substituting $\gamma = \frac{\alpha(1-\epsilon)}{C p_i}$ in equation B.10 we have,

$$\varepsilon_i = 1 - \frac{\frac{1-\epsilon}{\epsilon}\ln\left(\frac{1}{1-\epsilon}\right) - \frac{1-\epsilon}{C}}{1 - \frac{2-\epsilon}{2C}}$$

Simplifying this expression, we get

$$\varepsilon_i = 1 - \frac{2C(1-\epsilon)}{2C-2+\epsilon}\frac{1}{\epsilon}\ln\left(\frac{1}{1-\epsilon}\right) + \frac{2(1-\epsilon)}{2C-2+\epsilon}$$

Expanding the ln term gives us

$$\varepsilon_i \leq 1 - \frac{2C(1-\epsilon)}{2C-2+\epsilon}\left[\frac{\epsilon}{2}+1\right] + \frac{2(1-\epsilon)}{2C-2+\epsilon}$$

Which simplifies to

$$\varepsilon_i \leq 1 - (1-\epsilon)\left[\frac{2C-2+\epsilon}{2C-2+\epsilon} + \frac{(C-1)\epsilon}{2C-2+\epsilon}\right]$$

As $C \geq 1$ by definition, we have

$$\varepsilon_i \leq \epsilon \tag{B.11}$$

B.1.3 Case 3, $\beta > 0$

The analysis for case 3 is similar to that of case 2 but not identical. Here the profit density function is reflected with respect to the y-axis but the profit axis itself is not. We have

$$f_P(p) = \alpha + \beta p \quad \geq \quad 0$$

where β is a positive constant.

We also have, from the definition of the profit density,

$$\alpha \geq -\beta p_i$$

From equation 3.2

$$\varepsilon_i = 1 - \frac{\displaystyle\int_{p_i}^{p_{i+1}} \frac{p_i}{p}(\alpha + \beta p)\, dp}{\displaystyle\int_{p_i}^{p_{i+1}} (\alpha + \beta p)\, dp}$$

Simplifying as before

$$\varepsilon_i = 1 - \frac{\mathcal{N}}{\mathcal{D}}$$

$$\mathcal{N} = \alpha p_i \ln\left(\frac{1}{1-\epsilon}\right) + \beta p_i^2 \frac{\epsilon}{1-\epsilon}$$

$$\mathcal{D} = \alpha p_i \frac{\epsilon}{1-\epsilon} + \frac{\beta p_i^2}{2} \frac{\epsilon\,(2-\epsilon)}{(1-\epsilon)^2}$$

From equation B.1.3, we write

$$\alpha = C\beta p_i$$

where C is a constant ≥ -1.

Simplifying as before

$$\varepsilon_i = 1 - \frac{\frac{1-\epsilon}{\epsilon}\ln\left(\frac{1}{1-\epsilon}\right) + \frac{1}{C}}{1 + \frac{2-\epsilon}{2C(1-\epsilon)}}$$

Again, using the expansion for $\ln(\frac{1}{1-\epsilon})$ from equation B.6 we have

$$\varepsilon_i \leq 1 - (1-\epsilon)\left[\frac{2C - \epsilon - 2C\epsilon + 2}{2C - \epsilon - 2C\epsilon + 2} + \frac{C\epsilon - C\epsilon^2 + \epsilon}{2C - \epsilon - 2C\epsilon + 2}\right]$$

By definition, $\epsilon \in (0, 0.5]$ and $C \geq -1$. For this range of values for ϵ and C, $(1-\epsilon)\frac{C\epsilon - C\epsilon^2 + \epsilon}{2C - \epsilon - 2C\epsilon + 2}$ is always positive. Hence we have

$$\varepsilon_i \leq \epsilon \tag{B.12}$$

Equations B.7,B.11 and B.12 prove lemma 3. □

B.2 Piecewise Power Profit Distribution

The profit density function need not be continuos at a bin boundary, hence we consider piecewise power function distributions of item profits. This generalizes the case where the distribution function is continuos in the entire range of profits. It also generalizes the earlier analysis for piecewise linear functional forms.

Consider item profit distributions within a bin of the form

$$f_P(p) = p^l + C \tag{B.13}$$

where $l > 1$ and C is a constant term.

We compute the average relative approximation error for such cases and show that its upper bound is ϵ.

Lemma 4. *For a power function distribution of item profits within a bin i, where the bin profits increase geometrically, the average relative approximation error $\varepsilon_i \leq \epsilon$*

Proof. From equation 3.2, substituting the power form for $f_P(p)$ and simplifying the

resulting integral gives us

$$\varepsilon_i \leq 1 - \frac{l+1}{l} \frac{p_i^l(\gamma^l - 1) + Cl \ln \gamma}{p_i^l(\gamma^{l+1} - 1) + C(l+1)(\gamma - 1)} \tag{B.14}$$

where $\gamma = \frac{1}{1-\epsilon}$.

We would like the RHS of equation B.14 to be $\leq \epsilon$, the target error. This is true when the following condition is satisfied.

$$\frac{l+1}{l} \left[\frac{p_i^l(\gamma^l - 1) + lC \ln \gamma}{p_i^l(\gamma^{l+1} - 1) + (l+1)C(\gamma - 1)} \right] \geq \frac{1}{\gamma} \tag{B.15}$$

We are interested in the case where $\gamma \rightarrow 1$. Evaluating equation B.15 at the limit using L'Hopital's rule shows that this is indeed true and condition B.15 holds true. This gives us the proof for lemma 4. □

B.3 Second Order Exponential Profit Distribution

Consider a profit distribution within a bin that is a second order exponential form.

$$f_P(p) = pe^{-Cp} \tag{B.16}$$

where C is a positive constant.

We compute the average relative approximation error for such cases and show that its upper bound is ϵ.

Lemma 5. *For a second order exponential function distribution of item profits within a bin i, with positive linear slope damped by exponential decay and bin profits increasing geometrically, the average relative approximation error $\varepsilon_i \leq \epsilon$*

Proof. From equation 3.2, substituting the linear-exponential form for $f_P(p)$ and simplifying the resulting integral gives us

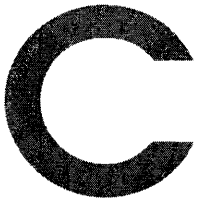
$$\varepsilon_i \leq 1 - \frac{1 - e^{Cp_i(1-\gamma)}}{1 - \gamma e^{Cp_i(1-\gamma)} + \frac{1}{kp_i}(1 - e^{Cp_i(1-\gamma)})} \tag{B.17}$$

where $\gamma = \frac{1}{1-\epsilon}$. As before, we are interested in the case where $\gamma \rightarrow 1$. Evaluating the limit using L'Hopital's rule shows that

$$\varepsilon_i \leq \epsilon \tag{B.18}$$

and this proves lemma 5. □

In summary, we have shown that for any linear, power or second order exponential distribution of item profits within a bin, the average relative approximation error as defined in equation 3.2 is never greater than ϵ. The computation of the runtime complexity now follows as shown in Section 3.3 in Chapter 3



Additional Empirical Results

This chapter contains supplementary numerical results to those presented in Chapter 4. Table C.1 shows the running times and observed errors for a set of uncorrelated problem instances with an acceptable error bound of 3%. Figures C-1, C-2 and C-3 show the respective running times, optimum values and observed error for this set of experiments.

Table C.2 shows the running times and observed errors for a set of weakly correlated problem instances with an acceptable error bound of 5%. Figures C-4, C-5 and C-6 show the respective running times, optimum values and observed errors for this set of experiments. The number of levels of binning m was set to 4 here as compared to $m = 6$ in the experiments presented in Table 4.3.

Exp. ID	Capacity	CPLEX(Opt)	CPLEX(t ms.)	MSKP(Opt)	MSKP(t ms)	Error	Speedup
1	255	2795	190	2739	90	0.020	2.11
2	101	2354	180	2338	70	0.007	2.57
3	203	2740	180	2704	90	0.013	2.00
4	113	2445	190	2409	80	0.014	2.38
5	127	2532	200	2522	80	0.004	2.50
6	145	2620	180	2583	80	0.014	2.25
7	169	2682	190	2658	90	0.009	2.11
8	290	2817	170	2739	110	0.028	1.55

Table C.1: Uncorrelated Problem Results with 3% Acceptable Error

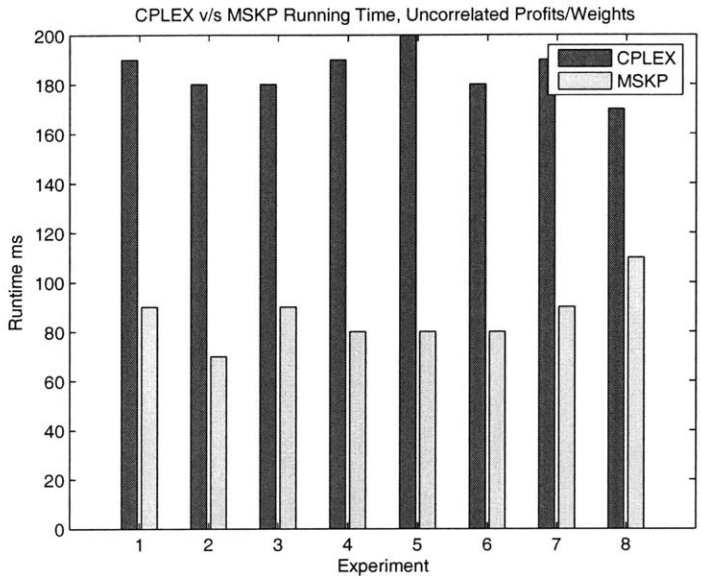


Figure C-1: Uncorrelated Problem Running Times with 3% Acceptable Error

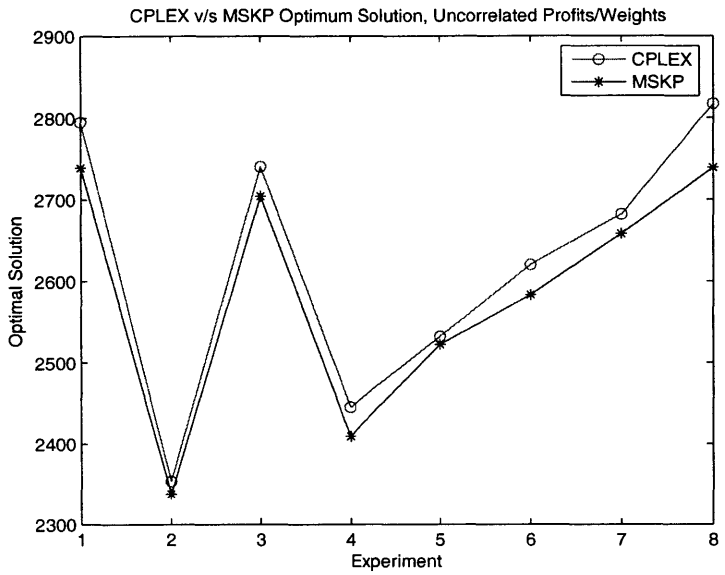


Figure C-2: Uncorrelated Problem Optimal Solutions with 3% Acceptable Error

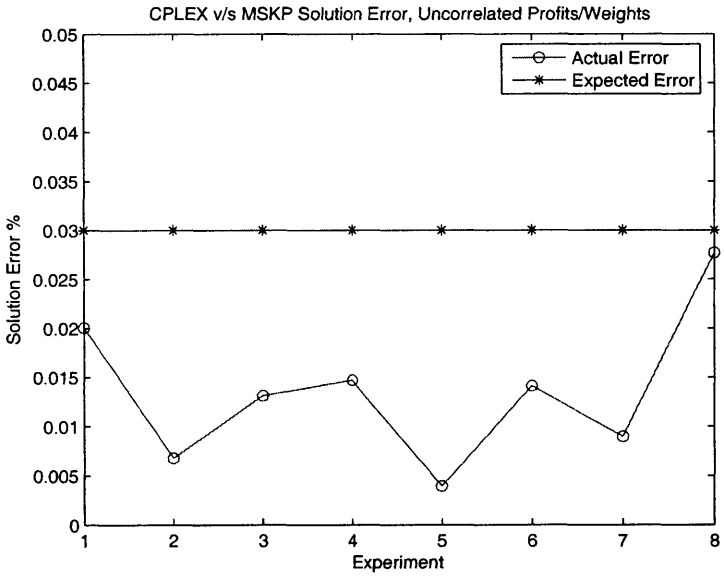


Figure C-3: Uncorrelated Problem Solution Error with 3% Acceptable Error

Exp. ID	Capacity	CPLEX(Opt)	CPLEX(t ms.)	MSKP(Opt)	MSKP(t ms)	Error	Speedup
9	100	244	195	237	130	0.029	1.50
10	113	268	190	261	130	0.026	1.45
11	127	294	200	289	130	0.017	1.54
12	152	340	160	339	120	0.003	1.33
13	169	369	210	364	130	0.014	1.62
14	191	406	200	402	120	0.010	1.67
15	203	425	190	411	140	0.033	1.36
16	226	458	200	439	60	0.041	3.33

Table C.2: Weakly Correlated Problem Results with 5% Acceptable Error

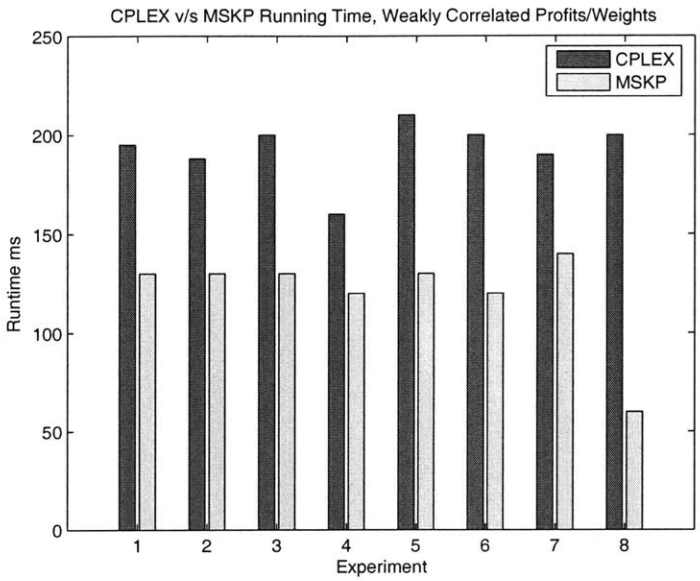


Figure C-4: Weakly Correlated Problem Running Times with 5% Acceptable Error

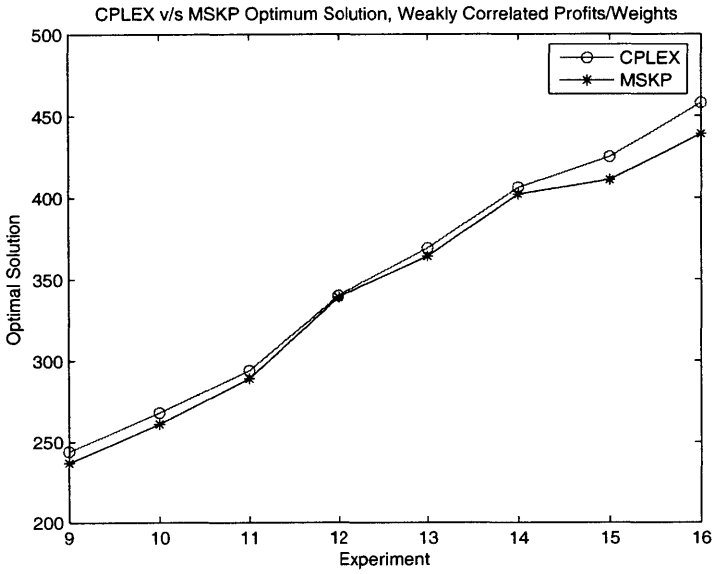


Figure C-5: Weakly Correlated Problem Optimal Solutions with 5% Acceptable Error

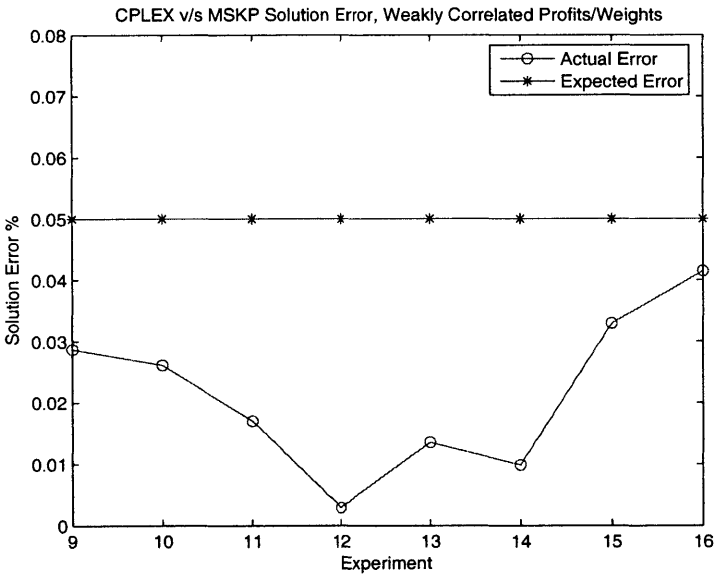


Figure C-6: Weakly Correlated Problem Solution Error with 5% Acceptable Error

Table C.3 shows the results of an experiment where the k value was varied for a fixed capacity and acceptable error of 3%. Figures C-7, C-8 and C-9 show the corresponding running times, optimum values and observed error for this case. The number of levels of binning m was set to 4 here. The relatively poor performance of $MSKP$ in the last six experiments is explained by the small choice for m and the increasing difficulty faced by the dynamic programming solution for $\epsilon = 3\%$ as k becomes smaller.

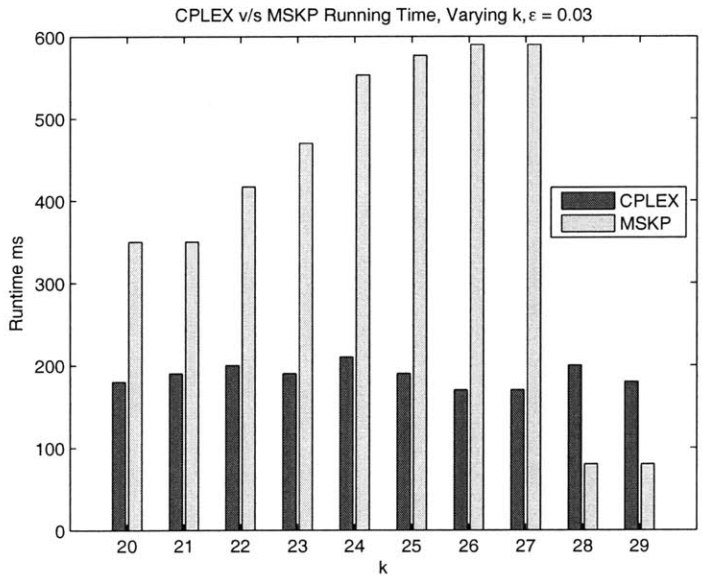


Figure C-7: Running Times for Varying k, $\epsilon = 0.03$

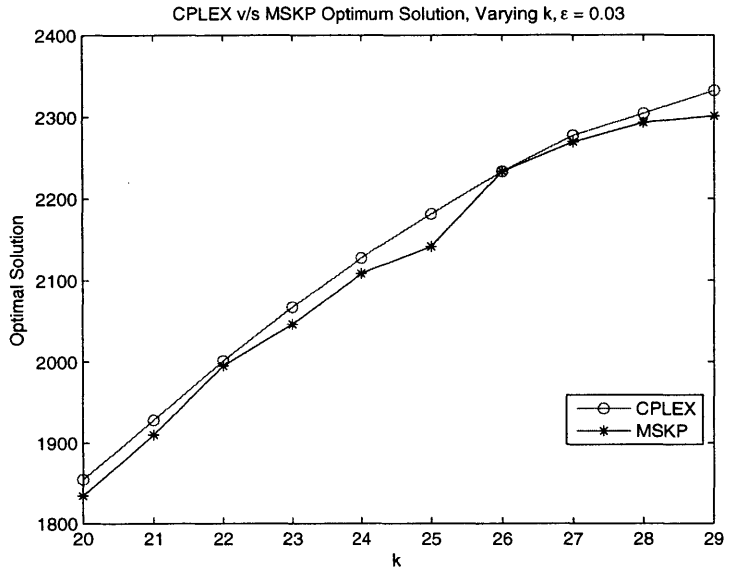


Figure C-8: Optimal Solutions for Varying k, $\epsilon = 0.03$

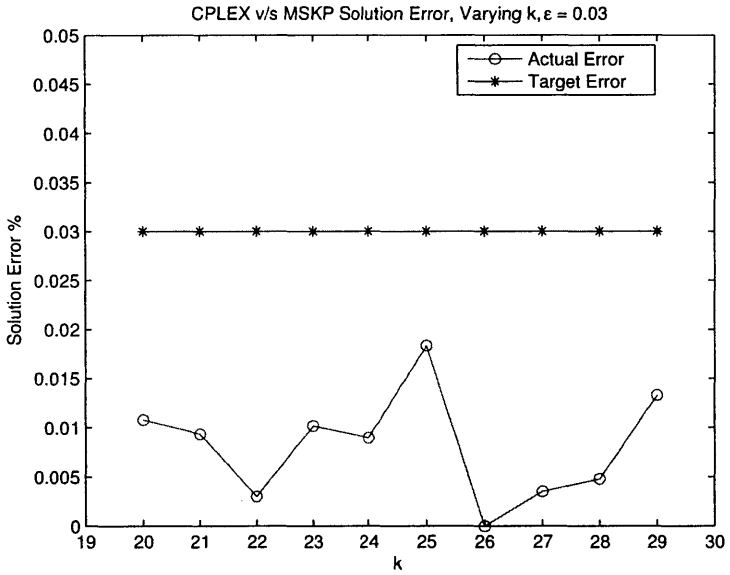


Figure C-9: Solution Error for Varying k, $\epsilon = 0.03$

ExpId	k	CPLEX Sol	CPLEX t (ms)	MSKP Sol	MSKP t (ms)	Solution Error	Speedup
201	29	2332	180	2301	80	0.013	2.167
202	28	2304	200	2293	80	0.005	2.458
203	27	2277	170	2269	590	0.004	0.288
204	26	2233	170	2233	590	0.000	0.288
205	25	2181	190	2141	577	0.018	0.329
206	24	2127	210	2108	553	0.009	0.361
207	23	2067	190	2046	470	0.010	0.426
208	22	2001	200	1995	417	0.003	0.504
209	21	1928	190	1910	350	0.009	0.572
210	20	1855	180	1835	350	0.011	0.495

Table C.3: Results for varying k with $\epsilon = 0.03$

Bibliography

[1] L. G. Babat. Linear Functions on the N-dimensional Cube. *Doklady Akademiia Nauk SSSR*, 222:761–762, 1975.

[2] Peter Barth. A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization. Research Report MPI-I-95-2-003, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, January 1995.

[3] Alberto Caprara, Hans Kellerer, Ulrich Pferschy, and David Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operations Research*, 123:333–345, 1 June 2000.

[4] Ismael R. de Farias Jr. and George L. Nemhauser. A Polyhedral Study of the Cardinality Constrained Knapsack Problem. *Mathematical Programming*, 96:439–467, 2003.

[5] Ramazan Demir. *An Approximate Dynamic Programming Approach to Discrete Optimization.* PhD thesis, Massachusetts Institute of Technology, 2000.

[6] Michael R. Garey and David S. Johnson. *Computers and intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[7] Mhand Hifi, Hedi Mhalla, and Slim Sadfi. Sensitivity of the optimum to perturbations of the weight or profit of an item in the binary knapsack problem. Technical report, Laboratoire de Recherche en Informatique d'Amiens, June 2003.

[8] Oscar H. Ibarra and Chul E. Kim. Fast Approximation Algorithms for the Knapsack and Sum of Subset Problem. *Journal of the ACM*, 22:463–468, 1975.

[9] ILOG. CPLEX 9.0. http://ilog.com, 2005.

[10] Hans Kellerer and Ulrich Pferschy. A New Fully Polynomial Time Approximation Scheme for the Knapsack Problem. *Journal of Combinatorial Optimization*, 3:59–71, 1999.

[11] Hans Kellerer and Ulrich Pferschy. Improved Dynamic Programming in Connection with an FPTAS for the Knapsack Problem. *Journal of Combinatorial Optimization*, 8(1):5–11, 2004.

[12] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems.* Springer-Verlag, 2004.

[13] Antoon W. J. Kolen and Frits C. R. Spieksma. Solving a bi-criterion Cutting Stock Problem with Open-ended Demand: A Case Study. *Journal of the Operational Research Society*, 51(11):1238–1247, November 2000.

[14] Bernhard Korte and Rainer Schrader. On the Existence of Fast Approximation Schemes. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 4*, pages 415–437. Academic Press, 1981.

[15] Eugene L. Lawler. Fast Approximation Algorithms for Knapsack Problems. *Mathematics of Operations Research*, 4:339–356, 1979.

[16] Michael J. Magazine and Osman Oguz. A Fully Polynomial Approximation Algorithm for the 0-1 Knapsack Problem. *European Journal of Operations Research*, 8:270–217, 1981.

[17] Silvano Martello and Paulo Toth. *Knapsack Problems Algorithms and Computer Implementations.* Series in Discrete Mathematics and Optimization. Wiley-Interscience, 1990.

[18] Monaldo Mastrolilli and Marcus Hutter. Hybrid Rounding Techniques for Knapsack Problems. Technical report, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Manno-Lugano, Switzerland, February 2002.

[19] Monaldo Mastrolilli and Marcus Hutter. Hybrid Rounding Techniques for Knapsack Problems. *Discrete Applied Mathematics*, Article In Press, 2005.

[20] Nimrod Megiddo and Arie Tamir. Linear Time Algorithms for Some Separable Quadratic Programming Problems. *Operations Research Letters*, 13:203–211, 1993.

[21] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, 1982.

[22] David Pisinger. Where are the Hard Knapsack Problems? *Computers and Operations Research*, 32(9):2271–2284, 2005.

[23] David Pisinger, Silvano Martello, and Paulo Toth. Dynamic Programming and Tight Bounds for the 0-1 Knapsack Problem. In *International Symposium on Mathematical Programming*, 24 August 1997.

[24] Marc E. Posner and Monique Guignard. The collapsing 0-1 knapsack problem. *Mathematical Programming*, 15:155–161, 1978.

[25] Sartaj K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1):116–127, 1976.

[26] Petra Schuurman and Gerhard J. Woeginger. *Lectures on Scheduling*, chapter Approximation Schemes - A Tutorial. 2007. Prelimnary version, to Appear around 2007.

[27] Francois Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, 86(3):565–594, December 1999.

[28] Vijay V. Vazirani. *Approximation Algorithms.* Springer, 2003.

[29] Gerhard J. Woeginger. When Does a Dynamic Programming Formulation Guarantee the Existence of an FPTAS? In *Soda '99: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 820–829, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.